DOCUMENT RESUME

ED 125 533 IR 003 617

AUTHOR Marcus, Richard S.; Reintjes, J. Francis
TITLE The Networking of Interactive Bibliographic Retrieval Systems.
INSTITUTION Massachusetts Inst. of Tech., Cambridge. Electronic Systems Lab.
SPONS AGENCY National Science Foundation, Washington, D.C.
REPORT NO MIT-ESL-R-656
PUB DATE Mar 76
GRANT NSF-SIS74-18165
NOTE 172p.

EDRS PRICE MF-$0.83 HC-$8.69 Plus Postage.
DESCRIPTORS Computer Programs; Computer Science; *Information Retrieval; Information Science; *Networks; *On Line Systems; Program Descriptions; *Research Projects
IDENTIFIERS CONIT; *Connector for Networked Information Transfer

ABSTRACT

Research in networking of heterogeneous interactive bibliographic retrieval systems is being conducted which centers on the concept of a virtual retrieval system. Such a virtual system would be created through a translating computer interface that would provide access to the different retrieval systems and data bases in a uniform and convenient way, even for the inexperienced user. An experimental interface, called CONIT, has been built to test the virtual system concept. Initial evaluation of CONIT, which connects four retrieval systems, suggests that the virtual system approach could be cost effective. Particular attention was focused on the requirements for a common command language, ease of use, and message interpretation and protocols in a networked interface. (Author/JY)

March 31, 1976

Report ESL-R-656

THE NETWORKING OF INTERACTIVE BIBLIOGRAPHIC RETRIEVAL SYSTEMS

by

Richard S. Marcus
J. Francis Reintjes

The research supported herein was made possible through the supported extended by the National Science Foundation through Grant NSF-SIS-74-18165.

Electronic Systems Laboratory
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

ABSTRACT

Research in the networking of heterogeneous interactive bibliographic retrieval systems has been continued. The concept of a virtual retrieval system has been studied. Such a virtual system would be created through a translating computer interface that would provide access to the different retrieval systems and data bases in a uniform and convenient way, even for the inexperienced user. An experimental interface, called CONIT, has been built to test the virtual system concept. Initial evaluation of CONIT, which connects four retrieval systems, suggests that the virtual system approach could be cost effective. Particular attention was focused on the requirements for a common command language, ease of use, and message interpretation and protocols in a networked interface.

ACKNOWLEDGEMENT

This report constitutes the final report for Grant NSF-SIS-74-18165, entitled "Research in the Coupling of Interactive Information Systems." The project was supported by the Division of Science Information of the National Science Foundation and covered the period July 15, 1974 through November 31, 1975.

We wish to acknowledge the cooperation extended to us by the National Library of Medicine, the Lockheed Corporation, and the Systems Development Corporation in conjunction with our use of their respective online retrieval systems.

We acknowledge the efforts of Mr. Joseph J. Passafiume, Staff Member in our research group, who, since he joined the project, has made a major contribution in the area of computer programming and systems analysis as related to networking.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 The Developing Information Transfer Scene

In the preceding 10 years there has been [1,2] a rapid development in techniques for achieving transfer of information among individuals representing a common community of interest, such as in a scientific discipline or technical field. Many of these new techniques have centered on, and depended upon, the rapidly growing technology of the digital computer, especially in its online, interactive, time-shared, and networked aspects. Thus we have seen the growth from experimental, to prototype, to operational stages of online computer-based systems that provide rapid simultaneous access, for dozens of users on widely-distributed terminals, to information in data bases containing up to 10^6 or more records with 10^9 or more characters.

For the coming 10 years we can predict with a fairly high degree of confidence that this trend toward systems of increasing capability will continue. Of course, one aspect of this growth will likely be an increased capacity for these information systems in terms of number and size of data bases and the number of simultaneous online users. Another aspect of development will undoubtedly be reduced cost; while the exponential increase in capabilities for a given cost in such computer system components as CPU, storage, peripherals, and data transmission that has marked the past 10 years cannot be expected to continue indefinitely, there is no indication that the rate of improvement for these cost factors will slacken in the near future. A third aspect of development that can be predicted with some degree of assurance for interactive information systems is improved computer-assisted instructional capabilities that will make these systems easier to learn and use by the average, non-computer-specialist user. A fourth area of development for these systems will likely be their continued refinement in terms of improved functional capabilities within the functional areas of the individual classes of systems: thus for example, retrieval systems may be expected to have more flexible search

and output capabilities.

The fifth and, perhaps, most challenging area for development in the near future is one that might be given the dual heading of networking and integration. Integration refers to bringing together for a user the many diverse information transfer functions. Besides bibliographic information retrieval -- where systems are now well developed for retrieval of references to documents -- there are now, at least in an experimental stage, many other capabilities, such as computer techniques for storing and retrieving numerical data and full-text alphanumeric information, alerting users on a periodic basis to new information that has entered a data base which is relevant to their profile of interest (selective dissemination of information -- SDI), identifying persons who can help answer questions and, generally, facilitating interpersonal communication. Other potentials for computerized information transfer services include techniques that facilitate "publication" (perhaps, entirely in an electronic medium), enable processing of retrieved information of all types and, finally, techniques that actually enable the answering of general questions posed in natural language or other formats and presentation of the answers in whatever format is most effective. Examples of various forms of presentation include natural language, numerical, graphic, oral, or combinations of these.

Systems of the far future may ultimately incorporate all these functions into one master information transfer system. The possibility of such a master system is one area for current research. However, for the near and intermediate future -- say, the next 10 or 20 years, it is likely that there will continue to exist separate systems for at least some of these functions. Therefore, enhanced user access to these separate systems through computer interfaces is another vital area that needs substantial attention at present.

Such interfaces are possible only in a computer network environment. Such an environment, if designed adequately, permits the interconnection of different systems. It also permits the interconnection

of different components of the same system, so as to make effective use of distributed computer-system components. The ability to do resource-sharing in a distributed computer network may well, then, be the key not only to increased effectiveness through functional integration but also to increased economy through efficient utilization of system components. The overall status of the general area of computer resource-sharing will be described next.

1.2 Status of Computer Resource-Sharing

The sharing of hardware and software resources in a single computer has been accomplished through the development of time-sharing systems like those pioneered at M.I.T.'s Project MAC [3] and elsewhere. With suitable digital communication links, such systems can extend resource sharing by providing access to users at remote locations over dedicated or switched telephone channels. A variety of systems software and hardware enables a user to select any program in the system. This program, in turn, can call on other programs to perform computation, transfer of data into or out of the system, and other kinds of processing. In the time-sharing environment users and programs can share these computer resources simultaneously.

Generally speaking, each computer program to be used in this kind of shared environment must be carefully designed to fit into the specific operating environment of a given computer and, in particular, its input/output characteristics must be well known to any using programs. Where these preconditions of cooperation and compatibility hold, the extension of the concepts of sharing to multiple computer systems and their associated resources is quite possible, although, of course, not without the resolution of substantial technical questions. However, a particularly vexing situation arises in the common case where one must contend with computer systems that have been independently and heterogeneously designed.

A partial solution to the problem of sharing resources from independent computers is found in terms of those telecommunications

networks which interconnect user terminals to different computers. One such network is that of the Tymshare Corporation (called TYMNET[4]) which interconnects users from a variety of terminals through "satellite" mini-computers to a dozen or more different computer systems. Networks of this type provide enhanced access to multiple, heterogeneous computer systems in that they enable terminals having different character sets and speeds to call a local telephone number (in most metropolitan U.S. areas and in some foreign areas) and get connected to widely dispersed and different computer systems. Thus, access is made easier in that the user does not have to contend with multiple telephone numbers and terminal connection protocols. Also, communications cost is lower in such a network than for separate direct-dialed or even leased-line connections, especially for the casual or infrequent user.

It should be noted, however, that terminal access per se is just one component of the process of sharing use of multiple, heterogeneous computer systems. At least two other components must be present for the effective sharing of heterogeneous computer resources. One is the ability of different computers and programs within the computers to transfer data to each other. A second needed component is the ability for either a program or human to make convenient and effective use of the various facilities once access itself is attained. In this regard, it is desirable that existing programs and systems be usable as building blocks for other programs and systems.

The interconnection of and transfer of data among heterogeneous computers -- including those having different manufacturers as well as differing operating systems -- has been an activity undergoing vigorous development in recent years. Several regional computer networks that can be mentioned[5] as examples of this development are: the Michigan Educational Research Information Triad (MERIT), the Triangle Universities Computation Center of North Carolina (TUCC), and the State of Georgia University System Computation Network. Perhaps the most well-known computer network currently in operation is that of the Advanced Research

Agency (ARPANET).[6] ARPANET is the prime representative of a class of networks featuring packet-switching technology. A commerical version of the ARPANET, call Telenet,[7] and developed by the Telenet Communications Corporation, has recently become operational. These networks provide the necessary uniformity and/or compatibility through hardware and software interfaces and communication channels and protocols so that data transfer and process control are enabled among the computers and programs.

Providing convenient access to the facilities within these networks has been the goal of a series of developments involving satellite minicomputers analogous in function to those mentioned above for the TYMNET network but attempting to provide more extensive and flexible capabilities. Many of these developments have been directly involved with improving access to ARPANET facilities. These include (1) the ARPANET Terminal Interface Message Processor[8] (TIP) developed by Bolt, Beranek, and Newman, Inc.; (2) the ARPA Network Terminal System (ANTS) developed at the University of Illinois; (3) the ELF "front end" system[10] developed by the Speech Communications Research Laboratory; and (4) the "Network Access Machine" (NAM) developed at the National Bureau of Standards.[11]

In addition to these attempts at providing more convenient access, many other developments have been taking place which seek to provide more effective means for the separately created and distributed computer programs to communicate with each other using the basic data transfer protocols so as to integrate for users the capabilities of dispersed resources. A few examples may be given to indicate the trend of these developments. Crocker et al[12] explained how protocols exist at different levels: low-level communications protocols are used by higher-level, "function-oriented" protocols whose primitives are more closely related to the substantive functions users require. Some examples of high-level protocols that have been developed for ARPANET use include (1) the TELNET protocol by which a user at a terminal controls a process in a remote host computer as if he were a local user of that host; (2) a File TRANSFER Protocol for transfering "raw" text files (means to

transfer structured files are currently under development); and (3) a Remote Job Entry (RJE) protocol.

Another major development in resource sharing has been the bringing together in one system of several different functions which get executed by invoking previously created programs on different computers. Such a system is the Resource Sharing Executive [13] (RSEXEC) developed at Bolt, Beranek, and Newman, Inc. RSEXEC is a distributed, executive-like system that enables ARPANET users to obtain, using a common command language, various services from different ARPANET host computers such as providing status information sending messages, and performing certain file-maintenance operations. A second example is found in a current project of the Advanced Research Projects Agency called the National Software Works [14, 38] (NSW). The purpose of NSW is to bring together within one system the means to generate and test computer programs so that, for example, a program can be written using an edit program on one computer, combined on a second computer, and run on still a third computer.

1.3 Problems of Utilization of Retrieval Systems

One area in which a sharing and interconnecting of computer facilities would be particularly useful is that of interactive bibliographic retrieval systems. It is in this area that the research reported on here has concentrated. As McCarn [15] has pointed out, uses of these systems have increased significantly in recent years. This application is thus starting to fulfill its early promise as one of the important applications to be served by the growing field of computer-based time-shared systems. Tens of thousands of searches are performed monthly by a number of different systems which have access, in the aggregate, to dozens of data bases containing, in total, more than five million references to documents of many different types -- e.g., books, reports, journal and news articles, etc. -- in a wide range of subject areas in science, technology, and the arts. There has been a steady rise in these statistics over the last few years as new systems and data bases have come online and more and more users have learned of their existence and retrieval effectiveness.

The very success of these systems has tended to aggravate the problem of convenient use because of the difficulties faced by users in learning how to interact with the multiplicity of heterogeneous systems and data bases. A potential user of different retrieval systems is faced with a series of obstacles right from the start: the necessity to discover these systems in the first place, to enter into separate procedures to gain access and reimburse costs, and possibly -- if the systems are not interconnected through a common network, as described above -- to make actual access via different terminals and separate locations. Other obstacles face the user once the initial access is made: different commands languages, retrieval functions, indexing vocabularies, and output formats. Even within a given system, access to different data bases is often frustrated by the differences in catalog record fields and indexing methods that the system may only partially compensate for. It is little wonder then, that currently access to these systems is primiarly through a professional intermediary -- a specially trained librarian, for example, rather than by the user himself.

It might be thought that a single system and data base should satisfy a given user. It has been our experience at M.I.T. with the Intrex [16] and NASIC [17] systems, however, that a single user generally needs access to many different bases, if not for a given search, then over a period of time as his needs change. Furthermore, in a community of professionals with heterogeneous interests, access to a multiplicity of resources pertaining to several disciplines is required. These resources are better stored as separate data bases rather than aggregated into a single huge data base.

These differences present substantial difficulties even to experienced users. In the NASIC at M.I.T. program [17], where librarians have been trained as information specialists to assist end users in searching online data bases, we have found that several weeks of training and continuing practice at the terminal were needed by the specialists

to get to a high level of proficiency and to maintain that level. A significant part of the learning difficulty was caused by the differences among data bases and systems. Even the specialists have found it desirable to specialize in a small number of data bases and, sometimes, in only one or two systems, at least partly for the reason of the heterogeneity of data bases and systems. Another reason, of course, is that existing systems have not yet realized the full potential of computer-assisted instruction.

In a study of current users of online bibliographic retrieval systems performed by the Systems Development Corporation under sponsorship from the National Science Foundation it has been reported [18] that a sampling of users surveyed by questionnaire indicated in the main that they were not having "major" difficulties in using different systems and data bases. However, over half of respondents did report "some" difficulty and the users surveyed constitute a biased sample in that they have already spent the effort to master the various systems and tend to be the heavy, intermediary-type users who would have less difficulty maintaining competence. Also, the results of this study are based largely on users' own evaluations without correlation with how well the user is operating the systems. In any case, a fuller evaluation of these recent results needs to be performed to see if it really is at variance with the more generally-accepted notions of difficulty as expressed above.

The end user may not have to master as many data bases and systems as the specialist searcher, but this contraction is more than offset by the fact that, in general, the end user has neither the time nor the inclination for training or practice. In fact, it is for this reason NASIC and others have decided that it is unrealistic in the present information-retrieval environment to expect end users to do their own searches, especially when the computer time -- as well as the user's time -- is such a costly commodity.

1.4 The Interface Approach to Connecting Systems

In order to investigate means to surmount the obstacles hindering convenient and effective use of the multiplicity of heterogeneous interactive retrieval systems, the M.I.T. Electronic Systems Laboratory has undertaken a research program to examine the feasibility of interconnecting interactive retrieval systems through computer interfaces. The computer interface would achieve compatibility among systems of heterogeneous hardware and software components through use of, or translation to and from, common retrieval protocols. (See Fig. 1.)

From its early stages our research program [19] has emphasized an approach in which the interface is, in effect, a common system into which and from which requests and results are translated automatically as they flow between user and serving systems. This approach has the virtue that a user attempting to retrieve information, when entering through the access mechanism provided by the common interface, sees a single virtual system in which all the complexities of the different retrieval systems and data bases are hidden and only a single uniform system is apparent. In this way the goal of convenient use of heterogeneous computer resources is achieved, at least for the particular application of interactive bibliographic retrieval systems. Two aspects of our approach that characterize our attempts at the application of networking are (1) the use of existing, major, stand-alone interactive systems without modification; and (2) an emphasis on serving the ordinary end user -- that is, a user experienced neither in computer programming, general computer usage nor in the use of interactive retrieval systems, in particular.

Our initial analysis [19] of the requirements for a common interface pointed to the need for three main kinds of logical components for an effective virtual bibliographic retrieval system: a common command language, a means for converting among indexing vocabularies and a common bibliographic data structure. Our review of these components, and of techniques likely to be useful in their implementation, is summarized below.

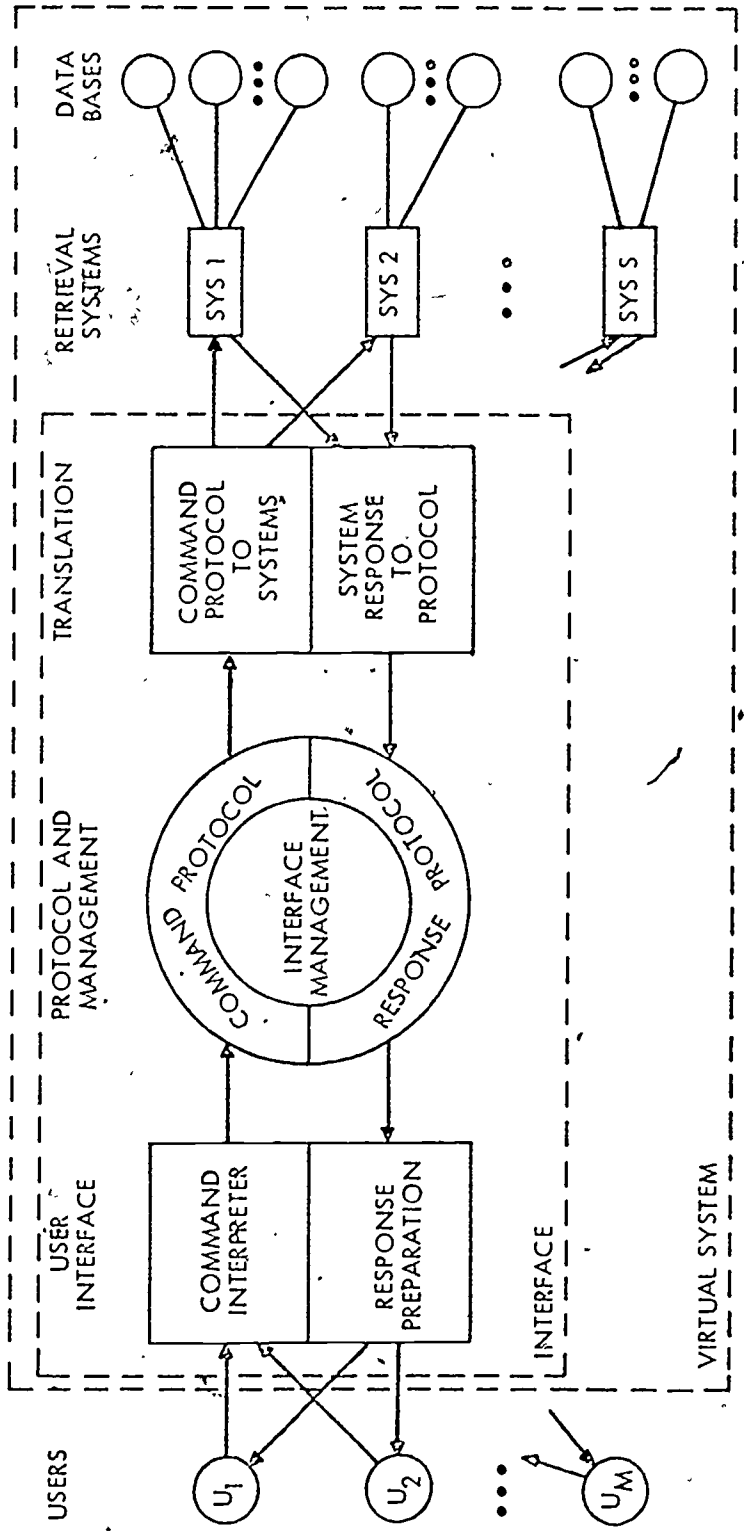


FIGURE 1 LOGICAL DIAGRAM OF VIRTUAL-SYSTEM INTERFACE

The Common Command Language

The common language should be a language in which all the functions for information retrieval operations can be conveniently expressed by users. One goal of such a language is to break the functions into the smallest components that find any different application in any two systems so that any function in any language can be expressed as a combination of common language functions, i.e., a macro function in the common language.

Indexing Vocabulary Conversion

We believe that a good basis for an intermediary language for indexing vocabularies is natural English. This is accomplished through a mechanism we have dubbed the Master Index and Thesaurus which contains the index and thesaurus elements of each of the data bases, including an ordered list of all vocabulary terms used for indexing together with the counts of the number of documents indexed by each and the thesaurus relations for each. (See Fig. 2.) In addition, through use of the techniques of phrase decomposition (that is, breaking a phrase down into its individual words) and stemming (dropping word endings so as to consider only the word stems) we can automatically identify most intervocabulary relationships.

A Common Bibliographic Data Structure

A common bibliographic structure can be based on the identification of data primitives or basic data elements, analogous to the basic component functions of the common command language. Data elements in any system can then be translated into, or composed from, combinations of basic data elements in the common data structure. The basic data elements would be hierarchically arranged into a data structure and, typically, the data element of a system would be equated to a higher-level node of the common data structure. An example of part of such a structure for data elements that relate to document contents and indexing is shown in Fig. 3.

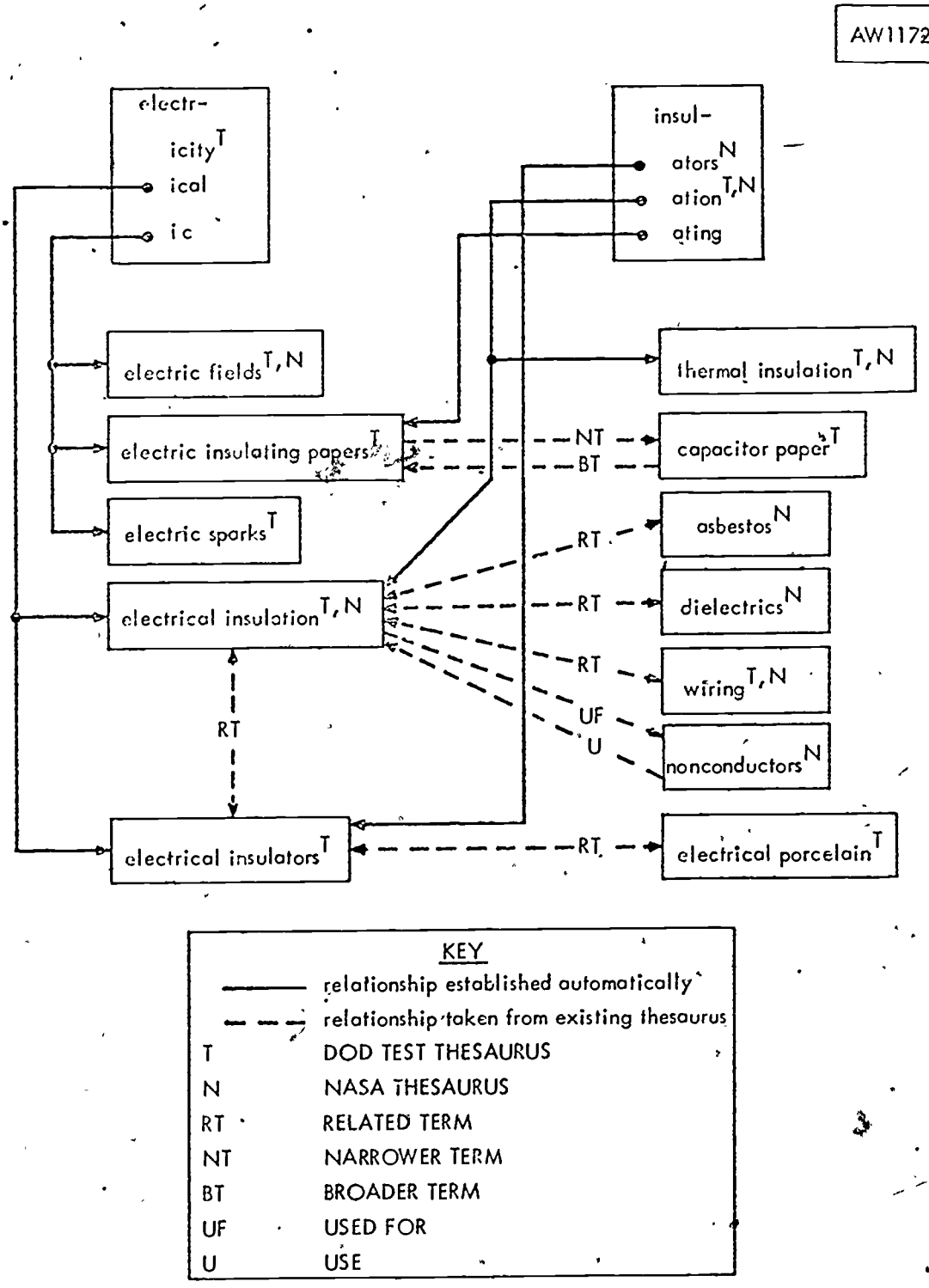


FIGURE 2 SAMPLE RELATIONSHIP AMONG TERMS AS MAINTAINED IN MASTER INDEX AND THESAURUS

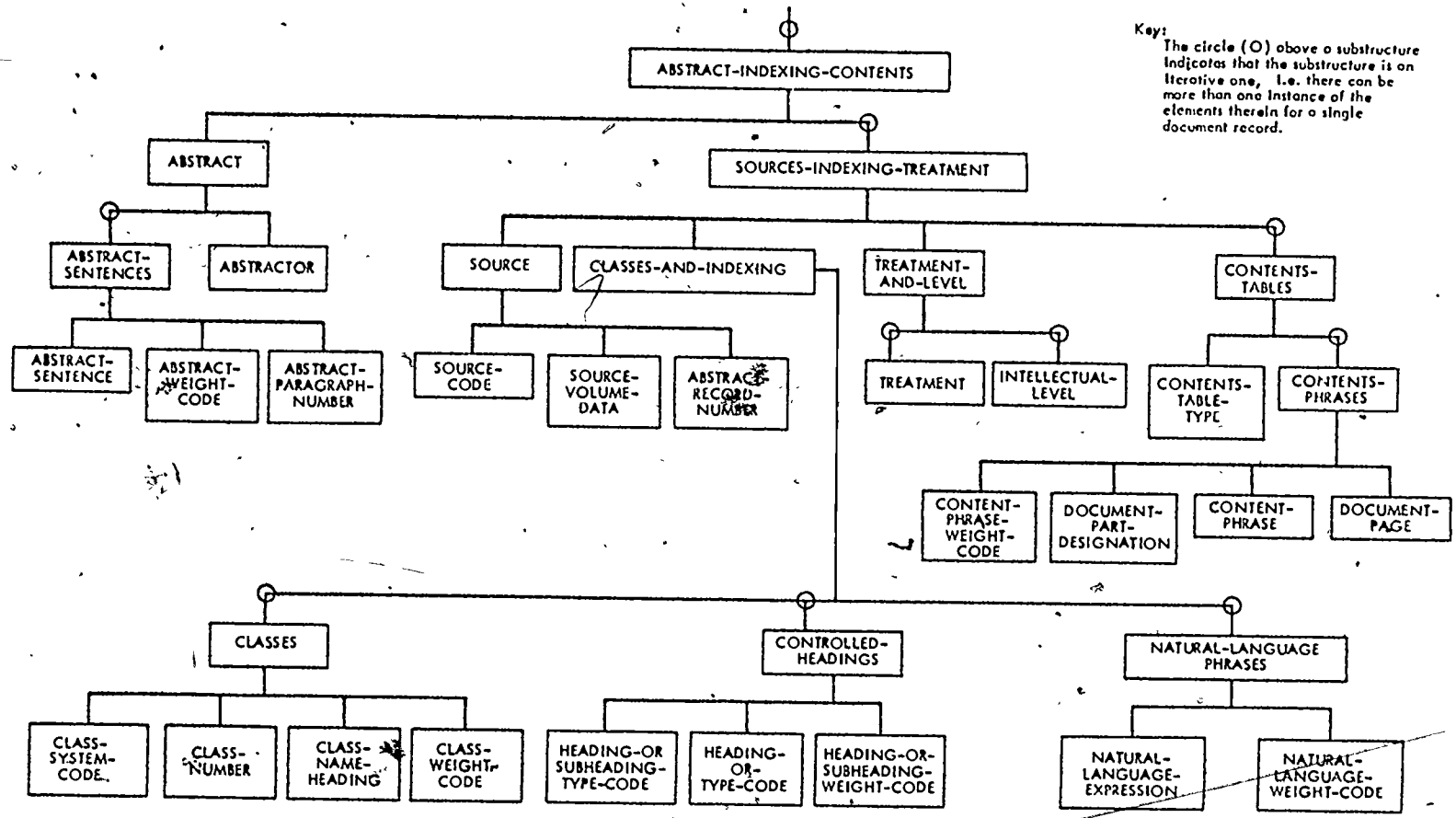


FIGURE 3 COMMON BIBLIOGRAPHIC DATA ELEMENTS AND STRUCTURE FOR THE INDEXING CATEGORY (INITIAL VERSION)

These and other aspects of the interface -- including how to fit the interface into the developing network framework -- will be discussed in the body of this report.

1.5 Outline of Work and Report

In this section we outline the contents of the remainder of the report and, in doing so, summarize the nature of the work that has been undertaken on the project, especially that portion that has been accomplished under the current grant during the past 16 months.

In Section 2 we describe the experimental interface that has been constructed on the M.I.T. MULTICS system in order to test the concepts and techniques developed in the theoretical component of our research program. At the beginning of this grant period we had a simplified experimental interface that connected to two retrieval systems containing about 8 data bases; a very simple translation of two commands -- a search and an output command -- was provided. During the present grant period the interface was extended to include connection to four retrieval systems with a total of about 50 data bases. Most of the foundations for a common command language was provided with a generally adequate translation to the four systems. A number of changes were made to the interface to improve the automaticity and reliability of establishing and maintaining connections to the different systems. In addition, a modest degree of translation of system responses to a common format was achieved and the beginnings of an instructional mode were implemented. In general, the experimental interface has now reached that point of development in which several (knowledgeable) users have been able to try it out for both demonstration and initial evaluation purposes.

In Section 3 of this report we list and explain those general principles for user/system interaction for online systems which serve as guidelines for our research program. Many of these guidelines had been developed by us and others prior to our current network effort but additional factors specifically relating to networking and to an

interface/virtual system were discovered and integrated into the general principles.

Section 4 includes a discussion of the general principles that could serve as a basis for the development of a common command language for interactive information systems and specific suggestions for the development of such a language. The advisability of using natural English as a command language for the interface is discussed. Here again, while the general principles for command languages have previously received considerable study, our work has extended them and applied them to the interface situation. We have tried to go beyond simply describing languages in the direction of prescribing optimized forms and explaining the reasons for the choices made.

In Section 5 we discuss the necessary elements for successful interprocess message communication among systems and human users in the interface situation. A model based on such investigations can serve three functions: (1) provide a basis for explaining some of the important features of interprocess communication in the general human/computer interactions and in the interface situation in particular; (2) provide a mechanism for detailing the actual interpretation and translation functions to be performed in specific situations; and (3) serve as a framework for software modules that would execute the interpretation and translation functions in a flexible, table-driven manner. Section 5 also contains discussion of how a common retrieval protocol might be relevant to the interface situation.

The experimental interface is described first in this report in order to make more concrete several elements of our work. However, the reader might well choose to concentrate on some or all of the analytic Sections 3 through 5, before Section 2, if he so chooses.

Section 6 gives our evaluation of the work to date. This includes a discussion of cost and benefits for interface systems of varying degrees of sophistication. Several side-benefits to work in the interface area are also described. Section 6 also discusses future work that could prove beneficial in the interface field.

References, and appendices follow in the remainding sections.

2. CONIT: THE EXPERIMENTAL INTERFACE

We have constructed an experimental interface on the M.I.T. MULTICS computer system in order to test concepts and techniques developed in the theoretical component of our research. We call this interface CONIT, an acronym standing for COnnector for Networked Information Transfer. In this section we shall describe CONIT in some detail so as to provide a concrete base on which the theoretical and evaluative studies of the later sections can be more readily understood. That is, in this section we describe what CONIT is; in later sections we explain why it is the way it is and how a better interface might differ from it.

It should be emphasized that CONIT is an experimental system and, as such, no attempt has been made thus far to provide a comprehensive interface. Rather it has been constructed so as to be able to test specific, representative functions and techniques. There are ways in which CONIT can be easily extended to cover more functions; other extensions would be more difficult. The nature of these extensions and their respective importance and difficulties will be discussed in this and later sections.

We shall first describe (Sections 2.1 - 2.4) how CONIT appears to the ordinary user, namely a person who might be using the interface to retrieve information for his own use from the networked retrieval systems and their data bases. Some indication of the software and hardware that underlie the interface will also be given. Later (Section 2.5), we shall describe the special features of the system which enhance its operation from the points of view of the analyst and designer.

2.1 Instructional Features

Let us start at the point at which the CONIT system itself has been called. (The initial connection and logging in to MULTICS and calling CONIT presents some special considerations that we shall discuss later in Section 2.5.5). Upon entering CONIT the user is made aware that instructions on how to use CONIT are available. The initial message (see appendix A) tells the user that he may go ahead and use CONIT if

he knows how or, otherwise, it tells how he may get instructional information.

In this first-level of computer-assisted instruction the user has one basic command, EXPLAIN, by which to request instruction. The EXPLAIN command has the syntax*:

explain concept

where the one argument, concept, is the name of a concept -- or a mnemonic abbreviation for the concept -- that CONIT is being asked to explain to the user. The concepts that can be explained are related to each other in a hierarchical fashion: the explanations for the general concepts list the names of more detailed concepts. The currently available explanations are shown in Appendix B. At the highest level is the concept explain which can be invoked by the command 'explain explain'** or by the simple synonym 'help'.

The command 'speak terse' will cause CONIT to abbreviate its dialog with the user. The command 'speak verbose' causes CONIT to return to the normal, lengthly dialog providing extensive instruction.

2.2 System Selection, Connection and Detaching

The most elaborate command, in terms of the mechanisms required within CONIT to implement it, is the PICK command by which the user can request connection to a retrieval system and can pick a data base in which to search. There are five systems to which CONIT currently makes a connection: (1) The M.I.T. Intrex system resident on an IBM 370/168 under TSO in Cambridge, Massachusetts; (2) the Lockheed DIALOG system on an IBM 360/50 in Palo Alto, California; (3) the System Development

*In this report we shall use underlining in examples of language constructions to indicate variable elements.

**In this report we shall use single quotes to bracket a character string that could be used in the command language; the two outermost delimiting single quotes are not part of the string itself.

Corporation (SDC) ORBIT system on an IBM 370/158 in Santa Monica, California; and (4) the National Library of Medicine (NLM) MEDLINE system for which there are two implementations to which we can connect: one on a 370/158 machine at the NLM Bethesda, Maryland headquarters (referred to as NLM/MEDLINE) and one on a similar machine at the State University of New York at Albany (referred to as SUNY/MEDLINE). CONIT currently supports a virtual-system type interface to these five systems; these five systems and several other systems can also be connected in a "transparent" mode, as will be explained below.

There are different modes of physical interconnection to these five systems and these differences are reflected in the operations of the PICK command. These physical interconnections have been previously described [19,20]. One mode of interconnection as shown in Fig. 4 is through the ARPANET TIP at the National Bureau of Standards to the NLM MEDLINE system in Bethesda. The other mode of interconnection requires a "patch"-type, manually-set connection between two manually-dialed phone lines: one between a Boston-area TIP and the patch box and a second between the patch box and a computer having access to one or more retrieval systems. This latter computer can be the M.I.T. 370 with the Intrex system or it can be a local TYMNET satellite computer which provides connection to the Lockheed, SDC, and the two MEDLINE systems through the TYMNET network. The NBS TIP/MEDLINE connection is generally maintained whenever the NLM/MEDLINE system is available. The patch connections are made on an ad hoc basis as needed for the experiments. Note that both that NBS TIP and the patch connections can be used at the same time so that two retrieval systems can be connectioned simultaneously. Also, we fully recognize that these low-bandwidth, terminal-oriented connections are far inferior to higher-bandwidth, inter-computer oriented telecommunications that we would prefer (see Section 6.1); however, they have proved sufficient to carry out our initial experiments on the higher-level aspects of the coupling of information retrieval systems.

To select a system the CONIT user types

pick system

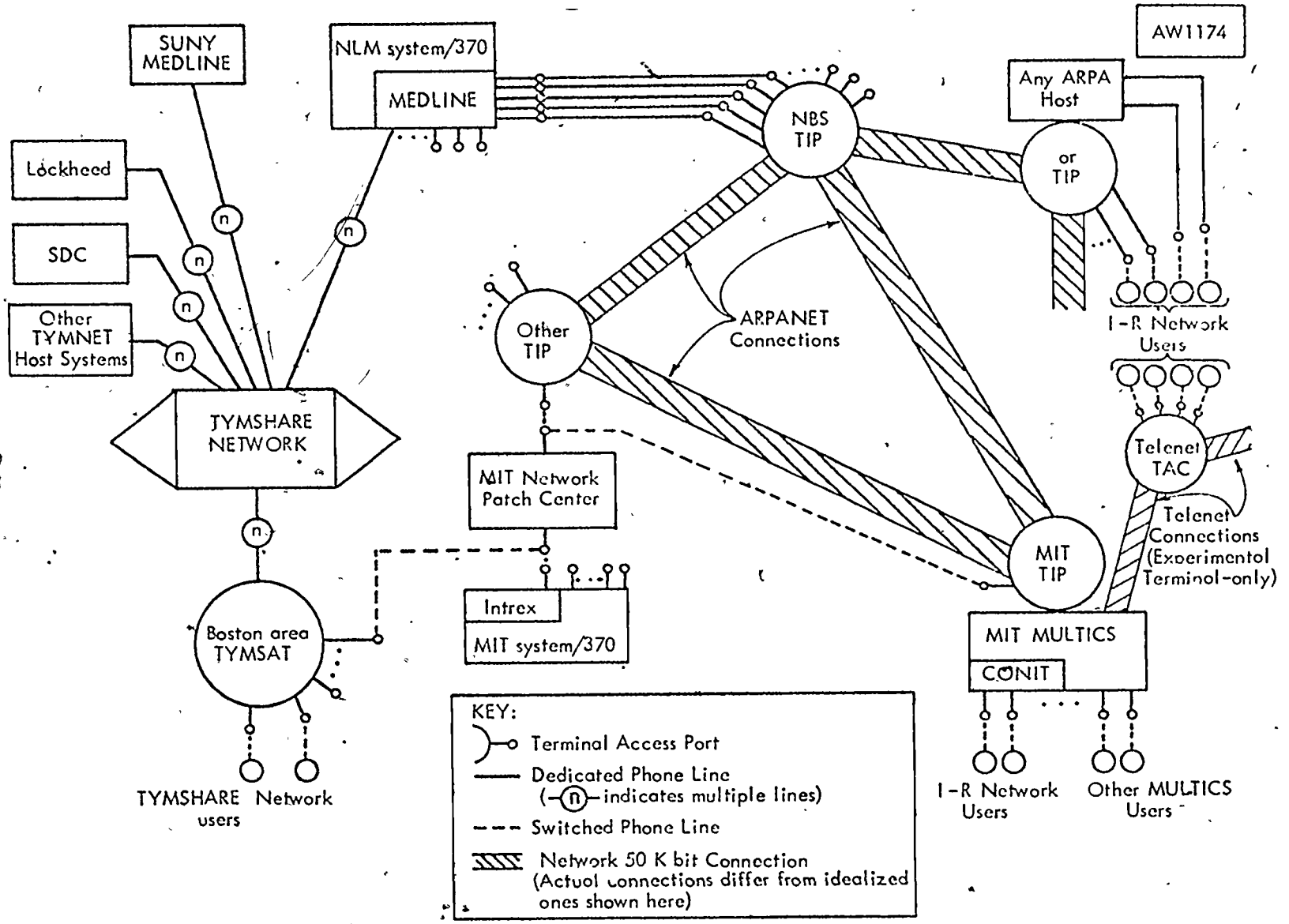


FIGURE 4 COMPUTER INTERCONNECTIONS FOR CONIT EXPERIMENTAL INTERFACE

where system is the name of the system. CONIT performs a number of functions in executing the PICK command (see appendix A for examples):

(1) Check to see if system is a valid system

(2) Check to see if system is already connected

(3) If system involves a TYMNET connection and there is a system already connected through TYMNET, then log the first system out. (The logoff protocol may involve the interchange of several messages to and from the first system)

(4) If there already is a system connected but it is connected through a TIP other than the one needed for the requested system connection, put the current system in a connected-but-not-active status and proceed to connect the second system.

(5) Establish a connection to the appropriate ARPANET TIP port if not already made. (This may require cycling through a number of TIP ports to find one that is available.)

(6) If system refers to TYMNET system, follow the appropriate TYMNET protocol to call up that system.

(7) Login by following the appropriate protocol. (This may include a separate call to the retrieval system after login; e.g., for NLM/MEDLINE).

(8) Answer any initial system questions (e.g., "Do you want experienced-or new-user mode?" -- CONIT works in experienced-user mode for compactness.)

When the appropriate response is not seen by CONIT (e.g., because of system failure or unavailability) in following one of the above protocols, CONIT returns control to the user with an indication of what the problem is. This indication may currently be of the most general kind (e.g., "proper response not seen") and may or may not leave the user in a position to continue to reselect another system.

2.3 Response Translation

As in all cases where response from a retrieval system is received by CONIT, there is a translation of retrieval system response

into a form more suitable to the user of the interface. There are two main mechanisms for implementing this translation. The first is a simple string-for-string translation table. The response message stream is scanned to see if any character strings match the "left-hand" or "input" or "argument" side of entries in the table. For each match found the matched string in the response stream is replaced by the "right-hand" or "output" or "function" side string of the matched entry in the translation tables. A separate translation table is active for each retrieval system connected to. See Appendix C for listing of translation tables. One function currently performed by these tables, for example, is to translate the string "PROG:", meaning in the ORBIT language that the message following is coming from the retrieval system, into the name of that retrieval system: whether SDC/ORBIT or NLM/MEDLINE or SUNY/MEDLINE -- the latter two MEDLINE systems being basically implemented in the same ORBIT framework as for the SDC system.

The second mechanism for response translation is simply the general one of the appropriate code within the routines that handle the dialog with the retrieval systems. For example, one function of these routines is to determine when any response is completed by looking for a specific "end-of-message" string, which is usually the "user prompt" i.e., "<NL>USER:<NL>" for Intrex. ("<NL>" stands for a new-line character or carriage return.) These system-specific user prompts are replaced by the CONIT common prompt "<NL>USER::<NL>" -- or simply "::" in TERSE mode. Many of the translations of both the table and the general routine mechanisms are, currently, simply to suppress a portion of the response (e.g., a system telephone number or the whole dialog about new or experienced users) or to pass along the message without modification to the user (e.g., broadcast news during login.)

2.4 General Retrieval Command Translation

The retrieval functions that can be performed through CONIT in the network of retrieval systems, besides the logging in and logging out described above, are largely accomplished, currently, through simple translations from the prototype common command language to the languages

of the individual retrieval systems through the mechanism of translation tables. These "user command" tables work in a fashion similar to the response translation tables. The command or request message stream as generated by the user is scanned, and any part of the stream that matches any entry in the command translation table for the system that is currently connected is modified by replacing the matched segment with the corresponding right-hand side of the table entry. This translated command is then sent to the retrieval system.

2.4.1 Data Selection

The CONIT user can find out what data bases are available in the currently connected system by using the command 'show data.' This gets translated to the commands '?FILES' in DIALOG and '"FILES?' in the ORBIT systems. No translation, as such, is made for Intrex but the mechanism is provided for such a request to evoke an instructional message explaining that Intrex has only one data base.

In the ORBIT systems, unlike DIALOG, not all data bases are available at the same time. The '"FILE?' command explains what data bases are available at the moment. To request a listing of all data bases that a system can make available at one time or another the CONIT command "show data all" is employed. This gets translated to '"EXPLAIN SCHED' for SDC ORBIT, '""FILES' for NLM/MEDLINE and '"FILES' for SUNY/MEDLINE. Note the small but crucial differences in the translations for 'show data [all]' even among the nominally identical ORBIT systems. Also note that the ordering of the rules is important; by insisting on a "longest-match-first" order 'show data all' takes precedence over 'show data' which takes precedence, in turn, over 'show' (see Section 2.4.5).

The command 'pick data database' is used to select a data base. The string 'pick data' is replaced by the string '.FILE' for DIALOG and '"FILE' for ORBIT systems. (Actually, the additional function '"USERS"' is added to the MEDLINE systems translations and '"TIME"' to the SDC/ORBIT translation both in order to make them somewhat more compatible with the DIALOG translation. The argument database,

which signifies the name of the data base to be connected, is left untranslated for the ORBIT systems. For DIALOG a translation is made from a mnemonic name to the number required by DIALOG: thus, for example, the strings 'eric' and 'ntis' are converted to the numerals '1' and '6', respectively. Of course, a user could use the appropriate numbers, if he knew them, and they would get transmitted to DIALOG without conversion.

The data base selection command takes precedence over system selection because the translations are executed before CONIT looks for commands it should execute rather than transmit. Commands to ORBIT systems initially required sending a final double quote (") and converting all lower-case letters to upper case. With recent modifications to these systems these requirements are no longer necessary.

2.4.2 Search Commands

The basic common search command 'find term' is translated '"FIND ALL term''select term' and 'subject term' in ORBIT, DIALOG, and Intrex, respectively. The 'ALL' argument to ORBIT indicates that all alternate meanings of the terms term are to be assumed desired instead of requesting the user to select some or all of these alternates. This translation is more in keeping with the intended meaning of the FIND command default option for the common command language. Actually, only the Intrex translation provides the automatic phrase decomposition and stemming that we wish to basic research mechanism to provide. (See Section 4 for additional details).

The more specific command to search for a particular author 'find author name' can be readily translated into DIALOG as 'select au=name' and Intrex as 'author name' but the translation to ORBIT '"FIND name (AU)"' is not possible with the current ranslation table mechanism because of the required rearrangement of the ordering of the 'author' and 'name' arguments. In the actual translation to ORBIT we use, '"FIND name', will work satisfactorily as long as the given author name is not also a subject index term.

The symbol '+' is the CONIT designation that, when appended to a character-string argument to FIND -- viz., 'magnet+', indicates a match should be made on any term exactly the same as the given string (e.g.,

'magnet') or any term having that string as a prefix (e.g., 'magnet', 'magnetic', 'magnetization', etc. This gets translated to the corresponding ORBIT symbol ':' or DIALOG symbol '?'. Intrex cannot handle this user-supplied stem; it takes words in the user-given terms and automatically tries to find the best stem to search under according to its stemming algorithm.

One could conceive of Boolean operations among the terms of a FIND command. The systems to which CONIT connect, however, are so dissimilar in their capabilities in this respect that CONIT currently makes only a minor attempt to take advantage of the potentialities in a common way. Intrex ignores all Booleans in the search command and relies on its Boolean ANDING of stemmed words; CONIT now does nothing to change a Boolean operator intended for Intrex, though it perhaps should, at least, issue a warning to any unwary user who tries to use an OR or NOT. ORBIT does allow a general Boolean capability within the search (FIND) command and these operators are passed along by CONIT to ORBIT as found. DIALOG does not provide for Booleans, as such, with its search (SELECT) statement; it does, however, provide some powerful "link" type operators for its "free-text" searching and one of these -- (F), as in 'term A (F) term B', meaning term A must occur in the same field as term B -- is taken as a reasonable equivalent for the CONIT AND operator.

The different kinds of search operations possible in the different systems, and the different manner of indexing for the different data bases in the different systems (or, even, within a single system) point up the inherent difficulty - and, often, impossibility -- of exact translation from a common language to existing retrieval systems and data bases.

2.4.3 Index Browsing Command

The CONIT command 'show index term' is intended to provide a display of terms alphabetically near to term in the index to the current data base. The translation is to the NEIGHBOR command for ORBIT and EXPAND for DIALOG. (Intrex has no equivalent command.) As can be seen

from the response and command translation tables an attempt is made by CONIT to make a common protocol for continuation of the index browsing function after the first display is made (5 terms for ORBIT, 15 for DIALOG). Thus "UP N OR DOWN N?" in ORBIT and the laconic "-more'" in DIALOG are both converted to "To see more type 'show more'.". Correspondingly, the CONIT 'show more' command is translated to the 'DOWN 5' command for ORBIT and '0' (page) command for DIALOG which both have the effect of requesting a second section of index term display equal in length to the first and continuing where it left off.

We may note, parenthetically, the difficulty of making these protocols exactly equivalent even for the simple case of length of initial section: either multiple commands would have to be sent to ORBIT and sections spliced together or the DIALOG response would have to be buffered and read out in sub-sections. This complexity would be compounded if we tried to incorporate the full capability of the ORBIT command with respect to a variable number of terms in either the forward (alphabetically) or backward directions.

We may note, also, that the full capability of DIALOG to tag these displayed terms (with "E and R numbers") and use only the short tags in the FIND (SELECT) command is implicitly available. The selection of multiple terms in this way is an implicit Boolean OR function. ORBIT does not have this capability; although, it could be implemented at the interface level at some programming expense.

2.4.4 Naming and Combining Retrieval Sets

The CONIT convention is to name the set of documents resulting from a search in the form: 'setn', where n is a number assigned sequentially for each new search set. This contrasts with the convention of using just a sequential number of ORBIT and DIALOG and the form 'sn' used by Intrex.

The CONIT language expression for combining sets takes the form:

combine setn1 bool setn2

where bool stands for one of the Boolean operations AND, OR, and AND NOT.

The conversion of this form to the appropriate retrieval system language is shown in the tables. Note that ORBIT and Intrex do not use an explicit command for the combine operation but rather use only the Boolean operator itself to indicate the function to be performed; therefore for these two systems, the translation for 'combine' is null.

2.4.5 Output Commands

To have CONIT display information about documents in some retrieval set the basic SHOW command is employed with the following syntax*:

show [mode] [setn] [fields] [docsj-k]

where,

(1) the variable argument mode stands for some special mode of output, e.g., offline.

(2) setn specifies a retrieval set

(3) fields is an argument string containing one or more data fields or field groups to be output; e.g., title, abstract, all

(4) The argument docsj-k specifies that output is to be derived from the catalog records of the j^{th} through the k^{th} documents in the search set.

We note again that particular features of several of the retrieval systems prevent a perfect translation to the several systems within the limitations of the current CONIT translation mechanism. Some examples of these difficulties may be instructive to the general problem of interface translations. Firstly, there may be no way of outputting some catalog data field for a given data base as implemented on a particular system. For example, the DIALOG system provides only a half-dozen or so fixed-groupings of fields for output purposes. For most DIALOG data bases, then, one cannot select for output just the author or just the title or just title and author, for example. The current translations simply make reasonable approximations. Thus, 'title' is translated to DIALOG output code 6 which includes the title and, variously, other citation information like order number, price, authors, etc. The default

*Elements in brackets indicate optional terms: they need not, in general, be included -- in which case they are supplied 'default' values by CONIT.

case (i.e., no data fields specified) is equated with the DIALOG code 2 output which is nominally citation information but often contains considerably more than that (e.g., index terms) -- in some sense there would be a closer translation to DIALOG 'title' category output, but getting the same result for title and citation output might cause confusion to a user. Note that even in these simple translations CONIT users can avoid the necessity to separate field names with commas as required by ORBIT.

The argument 'all' in CONIT is meant to indicate output of all fields is desired. This function has traditionally been performed by the argument 'FULL' in ORBIT. However, with the addition of abstracts to certain NLM and SUNY data bases (e.g., MEDLINE, SDILINE) this function is now performed by the argument 'DETAILED'. We may also have the situation in which the same function must be expressed differently in two data bases, even within the same system. Also, note how changes in the systems cause a translation to become incorrect.

Secondly, only the DIALOG system can provide the document selection function directly in the form given in the 'docsj-k' argument. A translation to ORBIT can readily be done when j=1, but the more general case requires the argument string 'm SKIP n', where m = k-j+1 and n = j-1. CONIT cannot perform this more general translation with its simple translation tables. Intrex cannot perform this document selection function within its output command. It can, however, perform the overall function by first creating a set of just those documents in question. Thus, the string of two commands

docs j-k/ouput

will perform the desired output. The problem is that the simple translation table mechanism cannot rearrange the fixed element 'docs' and 'output' and insert the variable elements 'j-k' between them.

When no argument is given by a user to specify the set it is assumed in the common CONIT language that the current (i.e., last-found) set is desired. The translation is implicit to ORBIT and Intrex which

have the same default arrangement. The translation to DIALOG is not now possible since that system has no default mode for the argument and CONIT does not yet have a way to remember the current set number. If no set number is given for DIALOG, CONIT now simply assumes set1.

Where an interrupt capability is available to the user it is anticipated that any good common language (see Section 4) will make the default condition on the document selection argument (docs j-k) of the output command be the whole set -- the user interrupting when he's seen enough. At present CONIT simply adopts the default procedure for the target IR system -- for ORBIT: the first 5 documents; for DIALOG: the first document (or first 5 for title only); for Intrex: the whole set.

In the ultimate common language the order of the arguments should be largely immaterial. Where this is true in the current IR systems (e.g., ORBIT and Intrex), the current CONIT language can accept that flexibility. Where a user is currently talking to DIALOG through CONIT he must accept the order stated previously: i.e., (1) mode, (2) set number, (3) field types, and (4) document selection. With the current translation table mechanism there is no way for CONIT to rearrange the order. The offline output function in DIALOG is accomplished by a different command (PRINT) than for online output (TYPE); therefore, the mode argument must be considered in conjunction with the show command name to determine the output translation. Also note that for DIALOG the user cannot now specify in CONIT the docs j-k argument without also specifying the fields argument.

2.4.6 Saving Output

A rudimentary capability exists within the current experimental CONIT for saving the results of searches from different data bases and systems in a common file created and stored by the interface and from which the user can display sections for subsequent online viewing. First a file is set with the 'name-file' (abbreviation:nf) command which has the syntax

nf filename

If filename names an existing file, that file is opened for viewing or appending to. If filename does not yet exist a new (empty) file with that name is created and designated for storage.

The command 'file' signals CONIT to append the response to the next succeeding command to end of the current saved file. Thus the sequence 'file' followed by 'show.:.' will cause the output of some search set to be stored in the saved file. The command 'view filename' causes the number of lines of text in filename to be reported to the user. Finally, the command 'lines j-k' causes lines j thru k of the current saved file to be displayed online.

2.4.7 News and Status of Retrieval Systems

Certain kinds of news and status information have been provided as parts of previously mentioned functions: e.g., broadcast news on login; database status on 'show data' commands; and timing information on login, logout, and database selection. The CONIT command 'show news' is the common means by which a user can request display of the standard news message from the currently active system. This command gets translated to '?news' for DIALOG; '"NEWS' for SDC, and '""NEWS' for NLM/MEDLINE. There is no translation to SUNY/MEDLINE, as such, but rather the evocation of a message (see explanation of 'sunynews') which explains that MEDLINE news can only be obtained from the NLM/MEDLINE sy tem. Note, again, differences among the several ORBIT systems.

2.5. Systems Analyst Functions

The CONIT functions we have described above have been those that make up the user interface, i.e., those communication components of the interaction directly used by, or seen by, an end user, i.e., a user whose main purpose in using CONIT is to find needed information from the data bases. We shall now describe those online interactive capabilities built into CONIT which assist a systems analyst to monitor, modify, and evaluate CONIT. Of course, some of these latter capabilities may be adapted to be useful to the end user as we shall indicate. These capabilities, together with those of the user interface and those

and those corresponding capabilities available to a programmer of CONIT's host system (MULTICS), make up what we might term the design interface.

2.5.1 Translation Tables

The command and response translation tables can be created, listed, and modified online. The command 'set_table [out] tablename' (abbreviation:st) causes a file with the name tablename[out] to be set up as the currently active table; if no such file exists, an (empty) one is created. If the optional argument 'out' is present, the table is taken as a response table and the file name is taken to be one with the string 'out' appended to the end of tablename (i.e., tablenameout otherwise, a command table is assumed. Command translation tables might be useful for implementing a "rename" feature for users (see Section 4.3.3.5).

To enter a rule in a currently active translation table one uses the command

replace [out] $matchstring=replacementstring[|]

(abbreviation for replace:rep), where matchstring is to be set as the left-hand, or argument part of the rule (see explanation in Section 2.3), and replacementstring is the right-hand, or function, part of the rule. Again, the optional argument 'out' is used when, and only when, the response table is intended to be modified. The optional delimiter vertical rule (|) is added after replacementstring in case that argument ends in a space character which would otherwise get discarded in the regular CONIT command-parsing operation. Note that presence of spacing characters in the argument and function strings can be extremely critical to the proper interpretation of a rule.

To list out online the current contents of a translation table, the command 'list_table [out]' is used (abbreviation for list_table:lt). In the listings of translation tables in appendix C the argument is the string between the left-hand margin and the equals (=) sign; the function is the string between the equals sign and the asterisk (*).

2.5.2 Dialog Modes and Language

Besides the argument pair TERSE and VERBOSE (see Section 2.1),

the SPEAK command can take other arguments to modify the form of the dialog. The argument 'monitor' can be used to cause CONIT to display the full dialog taking place among CONIT and the retrieval systems as well as the customary CONIT/user dialog which includes only a translated version of what the retrieval system communications were. Appendix B shows examples of MONITOR mode output, which can be very useful for debugging or demonstration purposes. In addition there is a mode evoked by the argument 'no_screen' (abbreviation:nsc) which causes CONIT to pass through certain formatting characters that are ordinarily "screened out" from retrieval system responses. The argument 'user' is used to get back from MONITOR to regular (USER) mode and the argument 'screen' is used to return to regular mode from NO-SCREEN mode.

The SPEAK command can also be used to go into a "transparent" mode of operation in which the command of the user are passed along to the currently connected system without translation and, likewise, the output from the system is passed back to the user without modification. The user is thus speaking the language of the connected (host) system. To enter this mode a user types the command 'speak conit'. Note that once in HOST mode the user can issue no instruction to be interpreted by the interface as such except 'speak conit'. All of these four SPEAK mode pairs are independent so there are $2^4 = 16$ possible mode combinations.

2.5.3 CONIT Status Reporting

CONIT can report the status of the current language, current modes, current host, patch connection, and TIP port use. The information is reported upon user issuance of the LIST_STATUS (abbreviation: ls) command which can take one of six arguments specifying the kind of information as listed below:

(1) 'system' (sys) -- the currently selected system and the other active system (if any).

(2) 'language' ('lang') -- the current language (i.e., CONIT only, for now, since user can't get this information in host mode)

(3) 'mode' - the currently selected modes such as VERBOSE or TERSE; SCREEN or NO-SCREEN; and MONITOR or USER.

(4) 'tip' -- the TIPs and ports currently being used.

(5) 'patch' -- the name of system currently connected to the patch

(6) 'all' -- information on all of the above.

One will note in the translation tables the rule 'ls all=ls all'. This is a current device making use of the longest-match principle for preventing 'all' from being translated as for the argument to the SHOW command.

Mode selection and status review as features for and users are discussed below in Section 3.4 and 4.3.7.

2.5.4 System and TIP Port Attaching and Detaching

The CONIT analyst can establish a connection to an ARPANET TIP port independently of whether or not there is, or will be, a connection made to a retrieval system over that port. The PICK command is used as follows:

pick tipname portnumber

where tipname is the name of some TIP (e.g., NBS, MIT) and portnumber is the number of the port to be connected on that tip. Thus, 'pick NBS 50' will cause CONIT to attach the user to one of the 5 ports on the NBS TIP that are regularly attached to the NLM/MEDLINE system, without forcing a login to MEDLINE as such.

CONIT also provides the facility for detaching an ARPANET TIP port connection by the command 'detach' (abbreviation:det). If the argument to DETACH is a retrieval system name, CONIT will detach (close) the connections to the TIP port through which the connection to that system had been made. Alternately, any TIP port connection can be detached by the commmand "detach tipname portnumber'.

2.5.5 Connecting and Disconnecting CONIT

CONIT may be evoked by issuing the command 'conit' at the MULTICS command level. To get to the MULTICS command level requires logging in

to MULTICS. This, in turn, requires (1) calling up the telephone number appropriate to one's terminal type, (2) setting up the terminal-to-telephone connection in data mode through a modem, and (3) issuing the 'login name' command followed by a password consistent with the personal name given in the name argument of the LOGIN command. MULTICS is fussy about upper-case/lower-case distinctions and the user must be careful, for example, to capitalize just the first letter of the name. Of course, the name used in the login must either be the official name for the CONIT directory (Conit) or some name which has access rights to that directory.

Users may connect to MULTICS through the ARPA and Telenet networds. That involves dialing a TIP (terminal interface processor - satellite minicomputer), establishing the terminal connection (including typing a character string identifying the terminal type, issuing a call to the MIT MULTICS computer ('o 44' -- i.e., open connection to computer 44 [MULTICS] for ARPANET, or 'c 617 mf' or 'c 617 ms' -- i.e., connect in the 617 telephone area to a MULTICS fast (1200 BAUD) port or to a MULTICS slow (300 BAUD) port, and then performing the MULTICS login procedure as above.

Users quit the CONIT program by giving the CONIT command 'exit' (abbreviation:ex) after which control is returned to the MULTICS command level. Any MULTICS command may then be given including 'logout' which disconnects the user from MULTICS. Disconnecting from ARPANET or Telenet is accomplished by breaking the telephone connection.

If the user has logged in to the CONIT directory in MULTICS he is captured by a "start-up executive command" program which automatically calls CONIT for him. Whenever the user leaves CONIT -- either voluntarily by the EXIT command or involuntarily from a system failure, the executive program automatically logs the user out of MULTICS. Access protocols that would be easier to use in network situations are discussed in Appendix D.

3. USER/SYSTEM INTERACTION: GENERAL PRINCIPLES

3.1 Importance of the User/System Interface

Online interactive computer systems are relatively new, having been in existence only about fifteen years. There is just now developing a body of literature [21-24, 35-37] which describes and evaluates features and facilities of the computer system which the user directly perceives as he interacts with computer. These system features and facilities include such system components as: (1) the command language; (2) the response dialog from the system; (3) "help" or other instructional facilities; and (4) user terminals. These system components are often known collectively as the user/system interface (or, simply, user interface). In our terms, the user interface is just one aspect -- the "front end" -- of the interface/virtual system which connects the user to retrieval systems through an interface system.

Despite the recent analytic work in the area of the user interface, there is, as yet, no agreed upon set of principles by which to measure or evaluate this critical component of interactive system. (The discussion in Section 1.3 emphasizes this point.) Needless to say, there are no existing, widely-known, operational online systems that are generally accepted as having anything approximating ideal user interfaces. In such a situation it is important that we attempt to describe the general principles which motivate us in this area and which, clearly, can strongly influence the nature of any analysis of translating computer interfaces for interactive systems. Such a description follows in this section.

3.2 Classes of Users

At least some of the controversy surrounding the user interface issue is, undoubtedly, caused by a failure to distinguish the several classes of users who may be engaging the systems. Earlier, in Section 1.3, we discussed some of the different kinds of users. One distinction that we made for retrieval systems was between an end user and an intermediary. The end user is a person who needs the information that is

derived from the data bases directly for his own work. The intermediary, who may be an information specialist acting as a delegated searcher, finds information for the sole purpose of passing it along to an end user.

Besides their classification by function, users need to be distinguished by their experience. Relevant experience comes in three categories. First there is the category of computer experience, expecially in regard to interactive systems and particularly with retrieval systems. Second is experience with the function to be served and the intellectual tools available to serve that function -- in our case the bibliographic retrieval function with the tools of bibliographic reference using knowledge of data bases, indexing and classification structure, etc. Third is experience with the subject matter of the data to be retrieved.

Thus, typically, the intermediary information specialist is experienced with the retrieval system and bibliographic search function, whereas the end user is experienced with the subject matter. Both classes of users are, in general, much less expert in the complementary areas. Of course, individual users possess varying degrees of experience in each of the three areas. The important point is that, in each of the three areas, the inexperienced user needs more help from the system than the expert user.

To date online systems, in general, have tended to be far from satisfactory for the inexperienced user; retrieval systems in particular have tended to work well only for an intermediary information specialist. One of our main goals is to consider what are the necessary prerequisites for system design by which the inexperienced user -- especially an end user -- can make effective use of online systems. Of course, a good system should train an inexperienced user how to become an expert user in time. Therefore, the good system should also allow for modes of operation that are efficient for the expert user and a mechanism for conveniently switching from beginner to expert mode at the user's discretion. In what follows below we try to outline some other general principles that support this basic one we have just described.

3.3 Instruction: Computer-Assisted and Other

Because a relatively large number of potential users of interactive information systems are inexperienced in one or more of the areas described above, it is very important to provide sufficient instruction to these users so that they can successfully take advantage of system capabilities.

There are several media for instructing users: (1) a personal medium in which human instructors teach system use; (2) a standard audio-visual medium including printed guides and manuals, slide and audio instruction, etc.; and (3) computer-assisted instruction (CAI) in which the computer itself is the basic medium by which assistance is given to the user. It has been suggested [23,26] that computer-assisted instruction is likely to prove by far the most cost-effective means for teaching system use. Of course, there can be combined media instruction as when the computer provides a real-time "hot-line" to a human aide or when the computer integrates and directs some audio-visual instruction. For example, an "online consultant" facility is available on MULTICS by which users can ask questions on their terminals and receive answers about the MULTICS system.[25]

In any case, our concern in our current work is primarily with what the interactive system itself can do to assist in the training and in otherwise aiding users in the use of the system. We shall outline in the rest of this section some principles pertaining to computer instruction.

3.4 Computer Techniques That Aid Learning

Two prime requisites for interactive systems are clarity and simplicity. The dialog from the system should be clear and easy to understand. Clarity requires succint, unambiguous expression of content for the individual messages, easily understandable format in which the messages are presented, and an ordering of the messages in a suitable sequence and structure so that user is led easily in a step-by-step fashion from his current state of knowledge to the desired conclusions. Information should be provided to the user at the time needed -- or, at least, with maximum probability that this should occur -- so as to

optimize its effectiveness. While the principle of clarity seems obvious, it may not be easy to adhere to. Opacity and ambiguity abound in interactive systems as they currently exist.

Simplicity is another cardinal principle that may appear obvious but is not necessarily easy to implement. Any complexity presented to the user will tend to confuse and, thus, inhibit successful use of the system. As system features multiply there is a tendency for the user/system interface to become more and more complex. Three avenues are available to the system designer to avoid unnecessary complexity: (1) design the whole system with careful foresight so that its elements and interrelations naturally form a coherent whole (including the design of instructional modes within the system); (2) apply the principle of clarity in instruction to simplify the explanation of the system (including the use of illustrative examples when appropriate); and (3) make the system modular with a simple basic core, as explained below.

A simple basic core means that the basic functions can be performed by using only a few simple commands. Only a few options are presented to the user; most options generally available from the system are hidden and take on default conditions. The user can extend from the core to other operations as he learns, at his own pace, what the other options are and how to use them.

Rapid response from the system is one requirement for online systems to be truly interactive. Generally speaking, delays in system response to a user request of more than 10 seconds cause confusion, frustration, interrupted train of thought, and other bad effects on the user. It is desirable that response times be less than 10 seconds and as short as possible -- although shortening times to less than one or two seconds may not be very useful. If the full request cannot be satisfied in a short time, it is often possible to start a response giving a partial answer within an acceptable time.

The user should be kept aware of system status, especially where rapid response is not possible. Just as indicators of the floor position and direction of travel of an elevator can make waiting for the elevator more bearable to the user (or can help the user decide not to wait any

longer), so too can the knowledge of system status relieve frustration and aid in control decisions for users of interactive systems. Of course, this principle must be balanced against the one of simplicity so that excessive and confusing information is not given.

To help the user make sure his wants are being correctly understood the system should feed back its interpretation of a user request as a preliminary response to that request. Thus, the system may indicate an obvious error in syntax or the user may detect an error undetected by the systems or a request otherwise undesired. Such feedback also acts as reinforcement to the user of correct system language and actions.

User control and flexibility in deciding what to do, and when, makes for optimum effectiveness of user/system interaction. The actions to be performed may be retrieval operations or informational requests. One kind of control that is extremely important for interactive systems is the ability for the user to interrupt the system, especially where (1) the system response is unacceptably sluggish (overlong response time); (2) the system response is overly lengthly (too much output); or (3) the user simply wants to change the direction or nature of the interaction without having to wait for the current operation to run to completion.

Flexibility implies an ability of the user, and the system, to pick among several modes of interaction according to the current class and state of the user and other context. A listing is given below of some of the more important modes that are possible. The modes are listed in mutually exclusive and opposing pairs and each pair may represent the choice, along one or more of the instructional dimensions dicussed above, of what degree of help for an inexperienced user, or user control for an experienced user, is desired.

(1) VERBOSE/TERSE. These modes relate to the length and and comprehensiveness of system dialog. There could conceivably be more than two modes along this spectrum, but it may be more important to switch among these modes for individual messages than to establish a whole third level.

(2) INSTRUCTIONAL/SERVICE. These modes relate to how much emphasis in the system dialog is put on instruction versus the provision of retrieval service as such. At one extreme there could be a completely tutorial mode whose sole purpose was to instruct. In general, there may also be a more or less prompting and other instruction given in and around service operations.

(3) INTERPRETED/STRICT. In a STRICT mode the system does exactly what the user requests. In the INTERPRETED mode the system goes beyond exactly what the user requested. For example, in a search in STRICT mode only the exact term as given by the user is searched, whereas in INTERPRETED mode an attempt is made to extend the search to terms related morphologically (e.g., as by stems) or semantically (e.g., as by thesaurus relations in a Master Index and Thesaurus). As a second example, in the translating interface situation a request that could not be translated exactly is, in STRICT mode, indicated as such to the user, whereas in INTERPRETED mode an attempt is made to find an approximate translation.

(4) AUTOMATIC/ASSISTED. In AUTOMATIC mode the system simply goes ahead and automatically does what it thinks best for the user, whereas in ASSISTED mode the user is allowed, and encouraged, to assist the system in making decisions. For the examples mentioned in (3) immediately above in the AUTOMATIC mode the system itself decides how to extend the search or make the translation whereas in the ASSISTED mode the system simply lays out for the user the options and lets him choose.

(5) HIDDEN/EXPOSITORY. How much should the system tell the user about what is going on? In the EXPOSITORY mode the system exposes a great many details (e.g., all the steps of a login process in connecting to a remote host through the translating interface). In HIDDEN mode the system assumes that the user shouldn't (needn't) be concerned with the details (e.g., simply report the success or failure of the aforementioned login process).

(6) VIRTUAL/TRANSPARENT. For the translating-interface/virtual-system approach a question is how thoroughly the virtual mode can be achieved as contrasted with making the interface be simply a transparent connector to host systems that the user must deal with in their own languages.

(7) INEXPERIENCED/EXPERT. For the inexperienced user all of the first-mentioned modes in the 6 above mode pairs are, ideally, chosen as a default mode. For the expert user either the complementary modes are chosen or the user is given the option of what modes he wants.

Of course, as indicated in the discussion of VERBOSE and TERSE modes, there may be many intermediate situations between the opposing modes in each mode pair. In the sections that follow we relate in greater detail the application of these principles of user/system interaction to the case of a translating interface to retrieval systems.

4. A COMMON LANGUAGE FOR RETRIEVAL

As discussed in Section 1.4, we are experimenting with the networking of retrieval systems in which a computer interface presents to the user a single virtual system based on a set of common features. Features that need to be put into a common form include the user command languages, the system response languages, the indexing languages, and the bibliographic data structures. In this section we shall discuss what specifications are appropriate to a common command language and, to a less extent, to a common response language, indexing language, and data structures.

Such common languages and structures can serve as a basis for a protocol by which distributed components of an information retrieval network may communicate with each other in a standard way. The networking aspect of the common language/protocol will be discussed more fully in Section 5. In this section we shall discuss the common language itself beginning with a critique of English as a possible basis for the common language.

4.1 English as a Common Language

4.1.1 Advantages and Disadvantages of English

It might be thought, at first blush, that natural language - i.e., English -- would be a good common language for interactive computer systems. English is widely used; it is the common language in this country and in many other areas around the world. In contrast to the requirement to teach an artificial command language before it is used, English speakers would not have to be taught English before using it as a language for conversing with the computer. English is naturally adapted to new conditions and uses. Finally, new developments [27-29] in the fields of computational linguistics and artificial intelligence have brought economic computer techniques

for handling natural language communications closer to achievement.

These comments about English are correct, as far as they go, but they do not present the full picture. Of course, on an international basis, there are many more potential users of interactive systems who do not speak English than who do. Thus, the correct argument is that natural language has the desired features of ease of use. English and the other natural languages do not, then, present a single universal language and different -- though perhaps similar -- computer routines must be employed to handle them as command languages. Also, while there are, undoubtedly important developments taking place in computer understanding of natural language, many of these are still in an experimental stage, and we do not yet have adequate cost-effectiveness data to predict accurately their success in our application.

Furthermore, the most important point to recognize, however, in considering English as a common command language is that general knowledge of the natural language does not, of itself, explain for a user what a computer system is capable of doing and what it is not. There is a vast disparity between the infinite variety of functions and requests that can be expressed in the natural language and still relatively very few and simple functions that interactive systems can perform.

One of the main problems for at least some users in making effective use of such systems is their lack of appreciation of the limited nature of system capabilities. The "super-brain" myth that views computers as all-knowing and all-powerful is one that continues to confuse inexperienced users. To tell a user, then, to "state your request to the computer in (ordinary) English" may be misleading in two ways: (1) it may foster the "super-brain" myth -- the user may infer that the computer understands (any) English as well as (or better than) a human -- and (2) it may postpone the necessary learning dialog between user and system in that the user feels the computer will always "do its best" for him without any special knowledge or guidance of the

system required by the user. (Note that this kind of misleading is, in fact, worse when the computer cleverly -- or perhaps, luckily - responds with what the user perceives as an intelligent response to his request.)

One could grant the difficulties of using English as a command language and still promote its use for the user as part of a learning dialog. This position has some merit. Thus, for example, a user request, whether natural language or not, can be analyzed fairly simply for keywords so as to select an instructional message that is probably relevant to the situation. (See Shapiro [30], for example). However, the use of English in an extended way to explicitly request detailed instructional information faces the same problems as for its use as a command language for retrieval functions.

The problems mentioned above, and the even more serious problems to be described below, can be alleviated by taking advantage of the interactive nature of the dialog: system and user can quickly converge on the proper understandings through a question/prompt-and-answer exchange. However, before we get into attempted problem resolution we should have a good appreciation of the detailed nature of the problems to be overcome so that we can better assure ourselves that the proposed solutions fully address the proper issues. It is in this spirit that we describe below some of the general problems of the use of natural language for command languages, and in particular in the information retrieval application, prior to our discussion of problem resolution which follows that description.

4.1.2 The Ambiguity Problem

The main problem with natural language is its ambiguity; that is, a statement can have many meanings. Usually, speakers can resolve these ambiguities sufficiently well so as to get along with each other at a rough level of understanding. This requires considerable mental capacity in terms of native intelligence, a large body of experience -- especially experience shared among the communicants -- and the extensive processing of linguistic data in context. When precise

communication -- of the kind we need for effective computer system operation -- is desired, however, a much greater requirement is placed on the communicants: they must either spend a considerable effort in conversational dialog so as to overcome the ambiguities for each case or they must have a mutually agreed upon precise language for communication on the particular topics under discussion. Therefore, we are faced with either an extra conversational burden or with the need to develop a more precise language like the specific, formal language that the natural language was supposed to enable us to avoid. However, counterbalancing these observations is the point that for a particular application we can build into the computer routines that interpret user statements taking advantage of a knowledge of the limited context implied by that application so that a fuller interpretion of user meaning is more readily determined.

Therefore, in order to evaluate these questions further and to make the above discussion more concrete, let us take particular examples from the retrieval application. Consider, for example, the question of naming and operating on sets of retrieved documents. Suppose that a user has performed searches using these three search statements: (1) "steel metallurgy"; (2) "steel castings"; and (3) "fractures in turbine blades." What, now, does a user say if he wants to find all documents that are in both the second retrieved set and the third retrieved set (i.e., the Boolean intersection of sets 2 and 3)?

First, one must realize that the user, based solely on his knowledge of ordinary English, does not necessarily know, nor will he necessarily use, any particular well-defined method of referring to a set of documents. There are, of course, many possible methods, several of which are actually used, as we have seen in Section 2.4.4. (For example, the n^{th} set can be referred to as "set n", "sn", or just "n".) Even more fundamentally, and making matters worse, the user does not necessarily know the concept of intersection or that retrieved sets are saved or that they may be operated on in other ways. Such a user

might use any of these referral methods; or worse, some combination of them; or much worse, an ambiguous circumlocation in ordinary English phraseology. For example, he might say:

(1) "Now everything on steel castings and fractures in turbine blades."

(2) "Get a combination of castings and fractures searches"

(3) "Can you show me citations on both the last two searches?"

If the first statement were given merely to perform the intersection it would represent a waste of user effort as contrasted with making a more concise statement using a specific "set" notation. Note that a statement like (1) could easily be interpreted to mean that the retrieval systems should perform a fourth search containing all the elements as given. Such an interpretation would be wasteful of computer and real time compared to combining existing sets; it could also result in a set different from the intended one depending on how the search match algorithm worked.

We may note that various ways of using the words "and" and "both", some of which are shown in the 3 example statements above, are generally ambiguous in English.[31] Thus "and" can be variously interpreted in the Boolean union (OR) sense ("We have a set of four eyes here: your blue eyes and my Brown eyes") as well as in the intersection (AND) sense ("Between the two of us the set of persons who are alive and have blue eyes contains just one member").

As an example in the searching application, consider the following four (perfectly reasonable) interpretations of Statement (1) taken as a search request:

(a) (steel castings) AND (fractures in turbine blades)

(b) (steel castings) OR (fractures in turbine blades)

(c) steel (castings AND fractures) in turbine blades

(d) steel (castings OR fractures) in turbine blades.

Sometimes semantic analysis and context clues can be used to help resolve ambiguities like these. However, automatic analyses may be complicated and costly, and often, in any case, the ambiguities can be resolvable only through questioning the user himself for his intent. We note here, in particular, that language expressions for search topics are not limitable in scope and context; they are, in this respect, unlike the commands themselves which are limited to the relatively few functions allowed by the system. This potential wide range of applicability of search topics is especially true in the situation we are dealing with: namely, a multitude of data bases covering many disciplines and document types, including those indexed under free-vocabulary as well as controlled-vocabulary (thesaurus) techniques.

Statement (2) could be reasonably interpreted as meaning either the intersection of the two given sets, as intended, or as a new search on just the two terms "castings" and "fractures". Also, it might take a fairly sophisticated algorithm to tell with any degree of assurance whether a word like "combination" in Statement (2) has a functional meaning or is intended as a term to be searched.

Beyond the naming and referring to retrieval sets, there are other problems brought up by the three example English statements. One problem is whether a search function or an output function is being requested. The natural language is ambiguous on this score. Of course, here again, syntactic and semantic clues may be used to help resolve the ambiguities. We may note, however, that even after it is determined that a search function is indicated, for example, other questions then arise: what is desired in the way of a matching algorithm, index elements to be searched, Boolean combinations, and data base searched?

Another problem that can be as thorny as the set-naming problem is the naming and specification of bibliographic data elements. Take just the term "citation", for example. It can mean (1) the references listed in the biliography at the end of a paper, or (2) the papers which

have references (in the first sense) to a given paper, or (3) the set of bibliographic elements by which one can "cite" a paper including the title, the authors, and, variously, such other elements as journal location, accession number, corporate author, editor, or other information to identify and access the document. How to refer to these various elements, either individually or in groups, is no simple matter, as this one example illustrates especially in the absence of a standard bibliographic data structure and nomenclature.

Finally, we may note that the third statement with its interrogative form could be taken simply as a request for information about how the system works or an implied request to perform the questioned operation.

4.1.3 Elements of English that are Desirable and Practical

The examples given above are clearly only a few samples chosen to illustrate the point; they could be easily extended to elaborate on the problems of the use of unrestricted and undirected natural language. Of course, besides those automatic analysis techniques alluded to above, answers to these problems may also be found, in part, at least, in those modes of operation employing directed, interactive instruction to the user on what is possible, and how to express it so that he may clearly and unambiguously specify the functions to be performed. One good way to instruct in the intricacies involved is to express them in a precise, unambiguous language, that is, at the same time, as simple as possible. Thus, because of considerations of effectiveness as well as cost, we are led in the direction of extensive instruction and/or a restricted subset of English that would avoid the ambiguities in expressing command functions.

The question then becomes one of determining which natural language elements can, and should, be included in a retrieval language for human use in the current state of evolution of computational linguistics. We have not fully resolved this question but we believe

such elements should at least include:

(1) English words as command-language vocabulary terms;

(2) "English-like" constructions for the commands -- i.e., having the "flavor" of, but not full variety of, English;

(3) English response to users (at least in VERBOSE and INSTRUCTIONAL modes);

(4) a natural-language approach to a common indexing and search vocabulary; and

(5) at least a minimal capability for transforming a natural language request into a suitable request for instruction on some system feature.

The above basic elements have already been demonstrated as being cost effective. Of course, how far one can or should go in releasing the restrictions on the formal command language and extending the variety of English construction and vocabulary that may be used is an important issue yet to be resolved. What our own analysis suggests is that the answer to this question involves consideration of such interrelated issues as:

(1) how much additional ambiguity is engendered in so doing;

(2) how much (if any) does the additional flexibiliy provide in terms of greater ease of learning and use;

(3) how effective and how costly are the techniques for automatically resolving these ambiguities (the answer to this question is one undergoing rapid change in an area of dynamic research and development); and

(4) how effective are the subsidiary interactive instructional techniques in handling the ambiguities not automatically resolvable.

In any case, how these five basic natural-language elements may be integrated with a more precise, formal command language is included in the discussion that follows of what a retrieval language should look like.

4.2 Desired Structure and Features of Interactive Languages

Having discussed the general principles of the user interface in Section 3 and some considerations on the use of ordinary English, or elements thereof, in the common retrieval language, we are now in a position to list those features that would make for a good language between computer and user. These general features are, to a large extent, we feel, independent of the particular application; the specific application to retrieval will be considered below in Section 4.3.

The command language for expressing requests of a computer system should be simple and clear, in keeping with both the needs of inexperienced users and the limited nature of what the computer can do. In the latter respect, it should be able to mirror the simple basic core and modular nature of the optimum user interface (Section 3.4) in that the command language subset required for the basic core should be very simple with complexity added only as needed to request the more specialized functions.

Let us assume the basic input mechanism for the user is a terminal with the ordinary typewriter keyboard containing at least the alphabetic and numeric characteristics and some punctuation. This assumption is made both because such input devices are now generally available and because it is not obvious at this point that any more elaborate devices (e.g., graphical input, light pens, function switches and buttons) can actually simplify the situation for the user.[32]

A simple command structure that lends itself to the above criteria is one having a command name followed by one or more arguments followed by a command terminator:

command-name argument-1 argument-2 ... terminator

The arguments are separated from each other and from the command name by simple punctuation -- e.g., one or more spaces. (The rationale for specific character choices will be given below in Section 4.3.). The number of arguments is variable; in the simplest case there would be no arguments at all. The command terminator, which in the simple case acts also as an end-of-message indication, is a single special (reserved) character -- e.g., carriage return -- which may follow the last argument, or command name, without any (other) delimiters.

Command names and arguments are primarily common English words, including numbers expressed as numerals. Common functions should not require the use of shift keys. The shift operation tends to be error prone and confusing for many users. Also, therefore, upper and lower case alphabetics should be generally equivalent. This command language terminology should be kept as unambiguous as possible. Thus, the same word should not be used with different meanings.

This last requirement can start to raise complications in special cases. For example, where free-vocabulary English is to be used in an argument, as in the search topic for the FIND command, there needs to be a mechanism for distinguishing a word in the controlled language terminology from that same word used in a free vocabulary sense -- e.g., author as an argument of the FIND command meaning either search in the author index or search in (any) index for the word "author". One mechanism, following the English convention, would be to enclose the word in quotes when used in the free-vocabulary sense - e.g., 'find title "author"' to request a search for the word "author" in the title index.

Further extensions to the language are needed to help users make the most effective use of the system. Pre-defined abbreviations for command terminology should be allowed. Any prefix of a pre-defined vocabulary term should be allowed as an abbreviation as long as it is not ambiguous with another term or prefix. (If such an ambiguous prefix were used the system should query the user on his intention.)

Beyond simple abbreviation the user should be allowed to use his own terminology by establishing synonyms for system language terms. Again, avoidance of ambiguity is the chief concern. One can also conceive of more complicated translations than just word-for-word synonym replacement. (See Section 4.3.3.5). In effect, the user should be able to construct his own dialect; the most advanced user would have a dialect with macro-like substitutions allowed.

The user should be able to string several commands together in one statement. This can be easily stated in the language if there is a command terminator distinct from the end-of-statement character. It may also be convenient to permit the use of special characters attached to, or connecting, arguments to indicate special functions like stemming and linking in search requests or editing (correction) of user input.

One major addition to the basic structure described above, which is needed to provide a convenient mechanism for stating relationships among arguments, is the subdivision of the arguments. Thus, for example, to indicate which documents to putput in the SHOW command, the elements "j" and "k" in the argument "docs j-k" are really subarguments to the primary argument "docs". (One might also say that "docs" is a subfunction or subcommand of the SHOW command). The more complete language structure then allows subarguments for arguments; sub-subarguments, and even deeper levels, are possible. It is desirable for simplicity to contain the logical depth of subarguments as much as possible. Also, to avoid complicating terminator requirements for subargument strings, it is desirable to make the argument structure apparent through the constraint on terminology in command and argument context.

Thus, the overall statement structure can be signified first by

$$C_1; C_2; \ldots, C_n.$$

where C_i is the ith command and the semicolon is used as command terminator. The command structure is represented as follows

$$N\ A_1\ A_{11}\ A_{12} \cdots A_{1m_1}\ A_2\ A_{21} \cdots A_n\ A_{n1} \cdots A_{nm_n}$$

where N is the command name, A_i is the ith argument, A_{ij} is the jth subargument to the ith argument, n is the number of arguments, m_k is the number of subarguments to the kth argument, and only one level of subarguments is shown. (No sub-subarguments). This structure is illustrated in Fig. 5.

Generally speaking, the interpetation of user requests should not vary with the reordering of arguments (or subarguments). Users should not be burdened with remembering some fixed order for elements, at least when they are essentially independent of one another -- like, e.g., the arguments to the output command. There are, of course, cases where order is important and may need to be preserved in the command language as where one was specifying a particular order for terms to be matched in a search request. Ultimately, as dialect-creating macros become sufficiently sophisticated, the expert user should be allowed to take advantage of ordering to shorten commands.

Users, also, should be generally free to give commands in any order they choose, as long as it makes sense. Thus, for example, a user should not be forced to scan an index display before making a search if he feels he knows what search terms he wants to use in the first place. There is a mode of operation of interactive computer systems in which the user is forced down a very particular path by, for example, having to "fill in the blanks." Thus, for example, the user may be asked what data base he wants to search and the only response he can make at that time is the designation of a data base. We do not advocate this respond-to-prompt-only mode because, first, it is so contrained and, second, even in the instructional situation for which it is often employed, this mode postpones demonstrating the command/argument type format which the user needs to make effective, individualized use of a system.

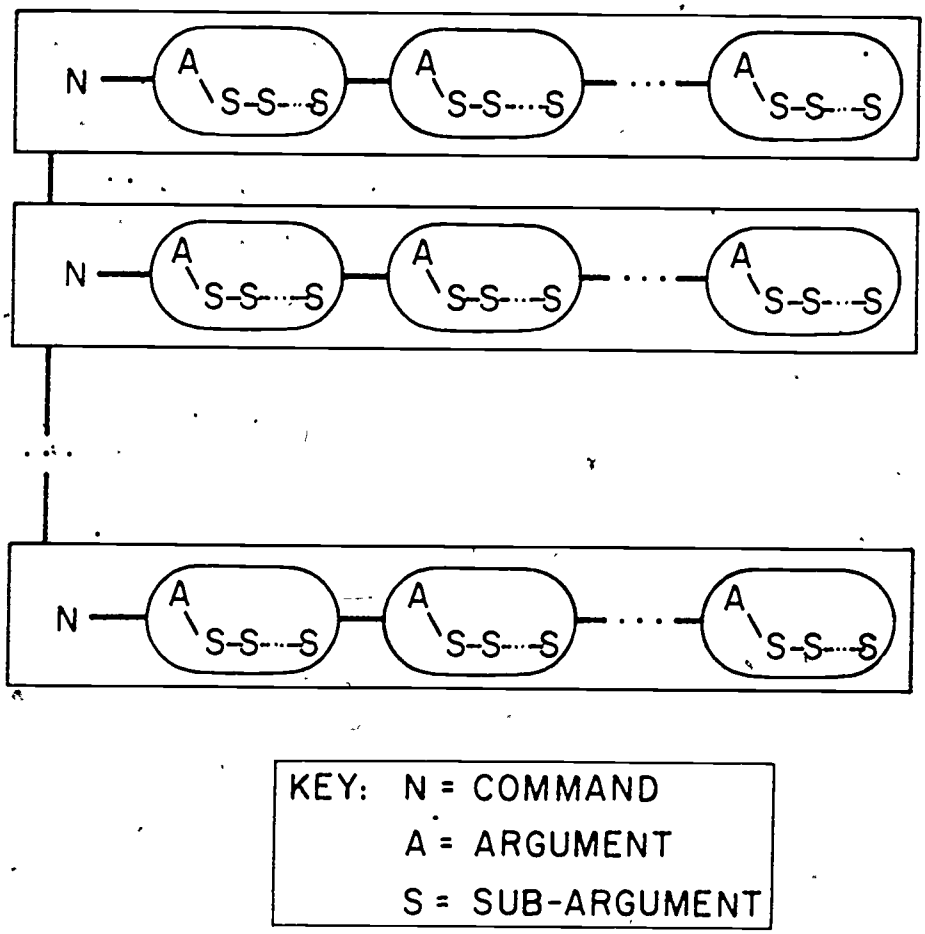


FIGURE 5 LOGICAL STRUCTURE OF USER STATEMENT

The system may, of course, ask a specific question such as "Do you want to see more output?". However, the user should be free to make a command other than in direct response to the question -- e.g., a new search request. Also, the question can probably be posed so as to prompt a response in the command/argument format, thus instructing the user in a more generally useful mode of communicating. For example, the output question above could have been stated as an imperative: "To see more output type 'show more' (or 's m' or 'sm')." Another advantage of this command/argument type instruction is that the user is being shown commands, that he may apply, without prompting in this and other situations.

4.3 Specific Plans for a Retrieval Language/Protocol

4.3.1 General Considerations

Having discussed the general structure and some features that make for a desirable interactive language, we now consider how these general ideas may be applied to the specific application of retrieval. We shall be extending, generalizing, and modifying the language framework of the CONIT experimental system described in Section 2 as well as trying to justify the choices made. The language described here will not be complete in terms of all possible retrieval functions or language specifications. While what we suggest here is incomplete and tentative, we are, at least making a start toward normative specifications for retrieval languages as well as raising issues surrounding the language question. We owe a debt to Martin[33] whose extensive descriptions of features of online retrieval systems has cleared the way for an attempt at prescriptions.

Some functions needed in the retrieval language may be very specific to the retrieval application, others less so. It is worthwhile to categorize this specifically into three levels. Some functions, like search and set combination, are quite particular to the retrieval application. A second class of functions, like initial connection to

the system and mode selection, are equally vital to a wide range of applications. The third class of functions, like editing (correction of user-statement typing errors), are, in the retrieval application, limited subsets of the functions performed more generally in another application (online text inputting and editing, in our example).

The reason for making this three-part classification is so that we can consider the interrelationships among the retrieval language specifications and specifications for other applications. The goal of integration of -- or, at least, standarization and compatibility among -- different applications as discussed in Section 1.1, impels us to consider these interrelationships. Thus, for the third class of functions we would want the retrieval language to be a subset of -- or, at least, compatible and consistent with -- any generally accepted, standard language with a more encompassing expression of these functions. In this case, hopefully, the subset will fall out simply from the larger set. Conversely, we would hope that the retrieval-specific functions of the first class could be simply adapted in other applications; the simple-basic-core principle should aid that goal.

Finally, for the second class of functions, we should try to choose specifications that are suitable for other applications as well as the retrieval one. Since there are no generally accepted, standard application (task-oriented) languages now, nor is there an accepted measure of consistency or compatibility among languages, we must expect our current attempts to be only tentative and subject to modification as developments in this area progress.

Another reason for tentativity is our uncertainty for a system to be implemented in the near future of exactly what functions would be included or how sophisticated they would be. Of course, the further in the future one goes, the cloudier the picture. Therefore, any language framework suggested at this point should be modular, flexible, and extensible.

We have previously distinguished the response language from the command language. Since the response language is largely in English -- especially in the VERBOSE and INSTRUCTIVE modes -- we need not be so critical about its form and structure so long as it satisfies the general principles of Section 3. Of course, the terminology should be consistent with and, indeed, be didactic for, the command language. In the command language discussion we shall make some comments on the content of the response language as pertaining to the command function under discussion.

We should distinguish various levels of language. The command language itself is what a user actually uses to issue commands. The exposition language is a metalanguage used to explain the nature of the command language, as to a user with the response language or to a meta-user analyst (e.g., a reader of this report). An internal language is a representation internal to the system of user commands, system responses, and other status information. There may be several internal representations as the commands and other messages are passed back and forth and each one requires, of course, a metalanguage of its own to describe it. The need for and nature of internal languages will be discussed more fully in Section 5.

The command language is conceived as being flexible and adaptable to user variations in a way not explicity indicated in the exposition of each command itself. Thus, upper and lower case variations in alphabetic characters and variable spacing before or after terms and delimiters as expressed by a user, are considered equivalent to the singlecase, single-space standard form. The exposition language may be different for user and analyst. For example, the user may have difficulty with metalinguistic devices like quotes. When the system says "type 'show more'" the user may wonder whether he must type the single quotes, especially where some systems do actually -- unfortunately (see Section 3) -- require quotes or other punctuation to distinguish one type of command from another. Therefore, we suggest de-emphasizing those kinds of metalinguistic devices for the user, as opposed to the

analyst. Metalinguistic devices that are better for the user would include those devices that would be less likely to evoke a user attempt to mimic; for example, examples to be copied or modeled by the user could be indicated by a different type font or color or a separate line with special indentation.

The device we have chosen of using capital letters to refer to a command name is a compromise that is not ideal because it suggests capitalization which should be avoided by non-expert typists. Our attempts at handling these problems may be seen by comparing sections 3 and 4 (intended for the reader analyst) with the dialog of Appendices A and B (intended for the user).

4.3.2 Retrieval Language Structure

4.3.2.1 Commands/Arguments/Delimiters

We take as a basis the "open" command/argument structure described in Section 4.2. The space character is taken as the delimiter between command names and arguments which are, at least for the simple basic core, common English words. This structure gives the simplicity and mnemonic value associated with simple, English-like phrases -- or, more exactly, imperative (verb) clauses.

Characters other than space (e.g., comma, period) are not nearly so "natural" in this respect nor do they separate terms so "clearly" to the eye nor are they as widely used in existing interactive languages. Any other punctuation or special characters (especially mixtures of different character types) and required abbreviations represent unwelcome complexity and mnemonic burdens to a novice or irregular user. The one problem with space is that it is a non-printing character and you cannot "see" it; where this matters -- e.g., where the space has not (yet) been followed by a printing character -- the difficulty is reduced if the input device has a good type-position indication. (Some of these considerations, especially as related to initial system access, have been discussed by Neumann.[34])

4.3.2.2. End-of Message Signal

Similarly, a carriage return provides a simple, easy-to-understand end-of-message signal. The ASCII new-line character is acceptable also if the carriage return feature is either included or added on (echoed) automatically. If a user statement should require more than one line to complete the user can cancel the normal effect of carriage return by preceding it with a special character, like hyphen - which is regularly used to indicate continuation from one line to the next in ordinary English. If a special device, like a function switch, is used to indicate end-of-message, it should cuase a carriage return to make it compatible with the simple case. It is clearly much less satisfactory to have special statement continuation devices depending on the particular command to be continued. In a well-designed system statements of more than one line should be needed only very infrequently. For example, long search phrases should be selectable as tagged elements from a dictionary display and the user should be able to break up strings of arguments and commands into shorter components.

4.3.2.3 Command Terminator

A good command terminator is semicolon (;). Several systems already use it as such and it has a corresponding meaning in ordinary English. Even this small degree of punctuation for delimiting is not desirable for inexperienced users. However, command stringing and, hence, the command terminator, is not necessary; commands may be issued on separate statements. It may be possible to eliminate the need for command terminators if command names are sufficiently distinct. However, the use of free-vocabulary index terms and the possibility of using arguments as separate commands (see below in Section 4.3.5) complicates the parsing problem and may make it inadvisable to try to avoid using the command terminator in command strings.

4.3.2.4 Bracketing

Delimiting or bracketing argument strings may become necessary

in some complicated situations like the nesting of Boolean operations (see Section 4.3.5). The parentheses could be reserved to handle those situations.

4.3.3 Dialog Control

4.3.3.1 Input Editing

The user needs to be able to change his input statement before he commits it to being sent via the end-of message signal. Two simple editing commands that cancel some or all of the preceding characters in the current unfinished statement should suffice for almost all situations. The delete-character command has the effect of deleting or canceling the last character entered by the user. The delete-line command cancels the whole line up to that point. Whether the canceled characters are actually removed and the type position reset, is a question of system sophistication.

Natural characters to use for these edit functions are the ASCII delete (DEL) and cancel (CAN) characters. However, current input devices may require shifting to get these characters. Also, current operating systems may require other implementations of these commands. Until these conditions are alleviated it may be better to accept other solutions. Thus, in the current CONIT we simply use the number sign (#) and at sign (@), respectively, required by MULTICS.

A simple extension of these edit commands is very useful. A string of n delete characters deletes the last n characters. An analagous extension could be employed for the cancel line command in multi-line statements. Note that this is one situation where the command terminator and other delimiting characters are not, and must not, be used in a command string.

4.3.3.2 Interrupting

As we have previously stated, the interrupt function is crucial to effective interactive dialog. Its meaning, generally, is to abort

(safely) the execution of the last given user statement and return control to the user to make a new request. If the user is still preparing his current statement, an interrupt would have the same effect as the cancel line(s) command but would also call for a user prompt.

In most existing time-sharing systems the interrupt is implemented using a special key -- the BREAK key -- which transmits not a character as such but rather a change in line condition (to zero state) for a specified period of time (say approximately 200 milliseconds). Such a signal cannot be transmitted through existing network connections without special hardware (although we have managed to fool at least one retrieval system host computer into thinking a string of null characters was break signal). Therefore, it is now becoming accepted practice in network situations to reserve a character in the regular character set to mean interrupt. For the common user command language such a convention should also be adopted and could be used in full-duplex operation. Note, also, that the interrupt command, unlike other commands, is used without waiting for a user prompt.

4.3.3.3 User Prompts and Status

The current CONIT user prompts -- "USER::" in VERBOSE mode and "::" in TERSE mode -- were mentioned in Section 2.3. Two colons are used because it is felt that a single character would too easily be ambiguous with other system response, especially in TERSE mode where it might be lost due to transmission or terminal timing errors, for example. The colon is chosen over other punctuation (e.g., question mark (?) hyphen (-), or greater than (>)) because it most clearly seems to suggest the notion that something is to follow. (E.g.; a question mark is often taken to have the significance; "I couldn't understand your last statement - please repeat or rephrase".) It is felt important that an inexperienced user be given more than just punctuation as a prompt that it is his turn. The word "user" in conjunction with the colons may have some advantages in suggesting whose

turn it is over other terms that may be used, such as "ready" or "type."

We have pointed out in Section 3 how user responsiveness and rapid feedback are essential to effective interaction. The user needs to know that the system and its components are working and that he can expect a response that is reasonably timely and worthwhile. Examples of kinds of status information that could aid a user in these respects are listed below:

(1) The terminal is working;

(2) The communication channels are open;

(3) The controlling system (e.g., the interface) is operational;

(4) Intermediate (e.g., network or operating) systems or target (e.g., retrieval) systems are operational;

(5) The user may now input a statement;

(6) The user statement has been
   (a) received,
   (b) interpreted successfully,
   (c) and this is its interpretation ...;

(7) the user request is now being actively worked on, or is queued up, by the interface and/or some other systems;

(8) For the current user request it will take so much longer (real time) to begin (or finish) a response at the following estimated cost.

To what extent, in what manner, and for what cost, the interface system and/or the other systems and components involved can, or should, determine and present this status information is certainly a large question which we only partially address in this study. For the end user in a highly virtual mode the amount and detail of such information should probably be highly limited. For an experienced

user and/or systems analyst the amount of such information might profitably be very much greater.

4.3.3.4 Verbose, Terse and Other SPEAK Modes

In Section 2 we described the VERBOSE (longer, more instructional) and TERSE (shorter) modes of the CONIT response language and how they are invoked by the SPEAK command ('speak verbose' and 'speak terse'). Two questions might be asked about the commands:

(1) Why use a command and argument format? Why not just two single-word commands; e.g., 'verbose' and 'terse'?

(2) Why the SPEAK command, in particular?

Since these questions are generic -- that is, they could be asked of many other linguistic decisions discussed in this report --, we shall spend some effort answering them here in hopes that these answers will also serve to help explain the general case.

The main reason for the command name with argument format is to help a user understand the nature of system functional capabilities through explicitness and consistency of format. This format mimics the verb-object/complement/modifier form of English verb-phrase structure. The command name is a verb used as an imperative to the system. The arguments complete or modify the imperative. Keeping this format consistently is worth something toward user understanding even though some additional number of words in response or command language may be needed. Advanced users can readily resort to a more compact form, if they choose. Thus, the one word 'terse' -- or even just 't' -- can be translated in a particular user dialect into 'speak terse'. Also, the system, in a somewhat more sophisticated parsing capability, can "understand" that an argument word, when used alone, implies the command word (unambiguously) associated with it. [We may note, parenthetically, the relatively greater difficulty of this kind of parsing if short abbreviations (t) are used in addition to fuller forms (terse)

because of the greater likelihood of ambiguity.]

The choice of terminology, as such, relates to several considerations. Short, common English words as suggestive as possible of the associated function(s) are what is sought. Brevity, of course, makes for simplicity, clarity, and ease of typing. Common English words are easier to learn and remember. Of course, the particular choice of words can be changed by a user for himself through synonym generation (renaming). Verbs are preferred for command names; nouns, adjectives and adverbs for arguments. (Of course, the most common words usually have more than one syntactical class, but one may predominate.) Commands, with their names, classify the functional capabilities available. Therefore, the choice of the word "speak" relates to a perception — that we would like the user to share -- that the user/system dialog may have many modes and the user should be able to select the mode by asking the computer to "speak" in a certain way. The terms "verbose" and "terse" were chosen as being somewhat more explicitly dialog-related than the other pair of terms often used for this purpose: "long" and "short".

Other dialog mode specifications would be made by other suitable arguments to the SPEAK command. For the command language itself we have used the arguments 'conit' and 'host' -- perhaps, 'direct(ly)' would be a better term than 'host'. System language names (e.g., "ORBIT", "DIALOG") and user (dialect) names (e.g., "smith") would also be allowed. Some other modes possible are indicated in Sections 2.5.2 and 3.4. Some of these modes might more naturally be set other than by the SPEAK command as, for example, by "'pick' or 'set' (conit) 'mode' 'automatic'." The language and parser should be at least as tolerant to a user putting these arguments variously with the related commands as it should be to ignoring the command name altogether.

The ordering of arguments in the SPEAK command, as elsewhere, should not matter. In fact, with initial default settings of 'verbose' and 'conit', the inexperienced user should not have to use the command

at all. A renaming macro should allow several modal arguments to be expressed in a single term, for example, 'speak myway', where 'myway' = 'terse conit expository'. If the user, then, implicitly asks for the same mode twice -- as in 'speak terse myway' -- the system should accept the redundant element with, perhaps, a comment on the redundancy in ASSISTED mode.

4.3.3.5 Renaming

In order to modify the language for his own purposes a user should be given a "rename" capability. One implementation of this capability is expressed:

rename oldword [as] newword

where newword, gets replaced by oldword by the system whenever it appears in the user statement. Note that this is a synonym-generating feature; the term oldword can still be used in its original sense. This may be contrasted with the situation where it is desired to revoke the meaning of some predefined vocabulary element (like 'and') so that it can be used in a different sense (e.g., as part of a controlled vocabulary term in a search). This latter capability can, perhaps better, be invoked by the "quote" mechanism mentioned above in Section 4.3.1 in which the original sense of a term is removed in each instance that it is preceded by (double) quotation marks. If it is desired to make the changed sense permanent, a different command should be used -- perhaps 'rename [and] drop oldword [to] newword', with 'change' or 'replace' being possible synomyns for 'rename [and] drop'. However, synomym generating and literal (quoting) functions are generally preferred over revocation in our virtual system approach because they allow users to fall back to, or more easily be encouraged into, using the common basic language vocabulary and, therefore, inhibit the development and use of incompatible special dialects.

The optional terms 'as', 'to' and 'and' may be useful in helping the user learn and remember the language construction at hand, expecially,

as in this case, where argument order does matter. Of course, the response language should be designed carefully to feed back the proper interpretation of what is being done. In any case, the use of these optional English function words to make the command language look more like English should be carefully weighed against the danger that such usage could (1) fool the user into thinking the computer understands English and (2) confuse the user by presenting him with additional vocabulary which the user might think, or suspect, that he is required to use.

The RENAME function can be extended, in stages, to permit the incorporation of multi-word terms and spacing requirements as in the CONIT REPLACE command, and finally, to a full macro capability. The more elaborate capabilities are certainly useful for a system designer (see Section 5); we shall not consider in detail here how important they might be for ordinary users and how much they might cost in terms of language sophistication. The macro translation capability may be symbolized in functional terms as:

$$g'[f_1(x_1),\ f_2(x_2)\ \ldots\ f_n(x_n)] \leftarrow g[x_1,\ x_2,\ \ldots\ x_n]$$

that is, a construction, g, containing variable elements x_1, x_2, etc., (along with fixed elements) is replaced by a construction, g', containing transformations of the variable elements.

4.3.4 System and Data-Base Selection and Connection

The use of the PICK command in CONIT to select systems and data bases was described in Section 2. It was felt that these two kinds of selections were sufficiently similar to warrant using the same command. It should not be necessary to use the argument 'data' since, if the final argument is not in a list of known systems, it may be taken to mean a data base.

The word "pick" was chosen because of its brevity and descriptiveness; other possible terms are "select", "choose", and "use". As was suggested in Section 4.3.3.4, the selection of various types of

entities could be done under separate commands (cf. SPEAK) or under a single command; in the latter case we have the just mentioned question of whether we need to specify the different types by special arguments like 'data', 'system', or 'mode'. The term 'data' is used instead of 'file' because it more specifically suggests the file to be searched -- i.e., the data base -- as distinct from other files that may be involved in the retrieval.

The PICK command actually incorporates two separate functions. The first is the selection of a system or data base for searching. The second is the actual establishment of a connection to that system or data base. For most situtions it may be satisfactory to perform both functions together. However, at times, as when one wants to avoid premature connection and extra cost, one may want to postpone the connection until the search is performed. For this purpose, these two functions might be separated at least by the interface system if not the user.

The selection of a data base may imply the selection of a system and the user should not necessarily have to perform, or even know about, the system selection. Sometimes the implication may be ambiguous (e.g., the NTIS and ERIC data bases are available through both SDC and Lockheed). In those cases, the interface may select which system to use, with or without the help of the user, depending on which mode was in effect. Of course, with the Master Index and Thesaurus concept, it is possible to select the systems and data bases automatically -- or partially so -- from the search topic itself.

In connecting to retrieval systems the login protocol is generally conceived as being performed automatically by the interface, as it is currently done in CONIT. However, in the more transparent (less virtual) situation the user may need to assist in the login procedure as with identification and password. Also, the interface user needs to gain access to the interface itself, probably through some login procedure. Therefore, it is appropriate to consider what might be a good

common login protocol, even though we might be forced to use other protocols currently. The general LOGIN command has the following syntax:

login system systemname id userid pass password

Several of these terms could be deleted if not required in a given situation (e.g., 'id' and 'userid', if no particular user identification is needed for a given system) or if the variable element implied the argument type determiner (e.g., a system name implying the 'system' argument). For security and other reasons modifications to this command procedure may be desired. Discussion on this point is given by Neumann [34] and in Appendix D where a more prompt-oriented protocol is suggested. In these discussions we note that 'login' may imply a 'logout' of the current system whereas a 'logout' by user means "stop and disconnect terminal" and 'exit' means "return control to calling system."

4.3.5 Search and Related Functions

4.3.5.1 Basic and Other Search Aspects

There are several aspects to searching that need to be considered in the retrieval language:

(1) System(s) to be used

(2) Data base(s) to be searched

(3) Kind of file to be searched

(4) Kind of data element(s) to be searched

(5) Matching algorithm to be used

(6) Elements to be matched

(7) Combinations of elements

(8) Type of results to be reported

(9) When to do the searching

(10) Naming of results

(11) Storing of (partial and full) results

(12) Sorting of results

(13) Effects on previous searches

The first question that might be asked with this large array of considerations concerns how many commands are involved: one or thirteen or some intermediate number.

Clearly, some of these considerations may be handled in separate commands before a search command, some in separate commands after the search command, and some by default. The only one that seems necessary to the search statement itself is (6): what one is searching for. However, it is desirable to be able to include any combination of the other 12 considerations within the search statement, if a user should so desire. A linguistic mechanism for so doing is simply to define separate commands for these functions which can also be included in a basic search command as we explain below.

The basic search command is 'find searchstring' where searchstring is an argument or argument string expressing what term or terms one is searching for. The word "find" is short, has good imperative search connotations, and is commonly used. The word "search" suffers from its ordinary usage in English: if you want to express that which you are searching for, the construction "search for x" is regularly used; on the other hand "search x" is normally understood in the sense "search in x" which in the retrieval application is best associated with the selection of a data base, or system, in which to search. The word "select" is not nearly as specific in connotation for the search function. We have explained in this report (Section 4.3.1) and in our previous report [19] why we feel there should be some explicit command name and not just 'searchstring' with the default command being FIND.

4.3.5.2 Selection of Data Bases, Files, and Search Elements

The default situation for system and data base searched would be the currently selected ones, unless otherwise selected automatically

through the Master Index and Thesaurus. The user could also make the selection within the FIND command as follows:

find [pick] [data] ntis radiation effects

Note that 'ntis' is at the level of a sub-subargument. The command/argument 'pick' -- and 'data', if possible -- should be optional here. The searchstring arguments are those arguments not otherwise identified as having pre-assigned meanings. The simple tutorial mode should suggest "PICKing" before "FINDing"; however, the "internal" selection should be allowed so that the user (1) is not forced to remember an ordering requirement and (2) can postpone the connection as long as possible if there is no separate "select-but-do-not-connect" function (see Section 4.3.4).

The kind of data elements to be searched may include elements like title, abstract, descriptor or identifier index terms, author, etc. These elements would be given assigned vocabulary terms which would be used as arguments in the FIND command to specify the elements to be searched. Thus,

find title descriptors neutron scattering

would signify a searc for "neutron scattering" in the title or in the descriptor index terms. The default condition (no data elements specified) is to search all data elements. (Our general philosophy is to be generous in retrieval; i.e., emphasize recall at the expense of precision -- on the theory that it is easier for a user to weed out the false drops than to appreciate what has been missed). The general question of what the common bibliographic data structure should be and how it maps into the structures found in existing data bases is discussed in Section 6.2.4.

There may be several kinds of files associated with a given data base. Searches are usually done on index (inverted) files. One may also search the full records of the data base. Because a full record search must generally be done sequentially, it is generally done

only on a small subset of the data base -- for example, a retrieved set. The index file has primary access data elements on which searching can be initiated and, sometimes, secondary data elements that can be scanned to determine whether a reference already found matches some secondary criterion. Thus, for example, a document reference found (through title index) searching, to contain two particular title words may be scanned for secondary information to determine if the two words are within a certain distance of each other and/or if the document referred to is of a given type, say book or report, etc.

The user may, to a certain extent, be shielded from these complexities. The user is instructed to express each search in the same form with each data element type argument within the FIND command preceding its searchstring. As long as there is at least one primary access data element, the search can be programmed successfully; otherwise, the user can be instructed to recast the search with at least one such term. Sometimes, however, -- as when a data element may be searched either as an index or a record search - it may be desirable to specify which kind of file is to be searched. For this purpose the additional argument 'record' may be inserted before the data element to be searched in the full record: thus, e.g.,

find descriptor radiation record title neutron

4.3.5.3 Term Selection, Combinations, and Matching

The argument searchstring may include a combination of terms satisfying a given Boolean relationship; e.g.,

find A and B or C and not D
find A and (B or (C and not D))

where A,B,C, and D are terms which must appear in the stated combination in each document matched. The order of operation is from left to right, unless countermanded by parentheses, as in the second example above. (Thus, the first example is parsed 'find ((A and B) or C) and not D'.)

This precedence order is preferred over one based on operator type because it is easier to explain to a user. (It might be noted parenthetically that the precedence order often chosen -- ANDing before ORing -- is opposite to the precedence more natural to the retrieval operation: first ORing synonyms for a given concept; then ANDing several concepts.)

The argument searchstring should not be taken to imply only an exact string match is desired; other matching algorithms are also warranted. Numercial data may be matched with arithmetic relations; e.g.,

year greater than 1970.

String data can be matched in various ways; e.g.,

find record abstract on?line:system:ℓ#ngu

where # means any one character (two # for 2 characters, etc.)

: means any number of unspecified characters

? means up to one unspecified character

More generally, we can describe a set of positional relationships, including relationships in word-oriented text; e.g.,

A within [exactly] ± n units B within ± n units C ...

where units can be character positions, words, lines, sentences, etc.

n is the number of units allowed from A to B

+ means B is to right (follows) A

- means B is to left (precedes) A

± (default case) means B may precede or follow A.

default for n is 0, i.e., A and B must be within the same (larger) unit -- e.g., words in a sentence

exactly means only the specified separation (nothing shorter) is allowed.

Note how the earlier string matching operation symbols are abbreviations. Thus, 'a##b' means the same as 'a within exactly +3 characters b'. A

very important ordering requirement is word adjacency: 'A within 1 word B'. A convenient abbreviated form to express this is 'A-B', where the hyphen carries over its natural language linkage signification. Lower order units take precedence in the ordering of the execution of retrieval operations over higher order units which, in turn, take precedence over simple Boolean combinations.

As previous work has shown,[16] retrieval results in general better than for exact matching can be obtained on word-phrase matching when (1) word order is ignored, (2) common words are excluded, (3) only word stems are matched, and (4) the Boolean AND is assumed. Thus, a good system will take

find economics of computer communications

and automatically set up a search to match on all documents having all three of the stems "econom:", "comput:", and "communicat:" in (any of) the index terms. Sometimes, however, better results can be obtained with a somewhat different algorithm. The user needs to be able to specify the variations. The 'within' argument string provides specification for word order. The phrase

exactly (A B C)

which may be abbreviated '(A B C)!', specifies an exact match is desired. A user-given stem may be expressed with the colon convention; e.g., 'find computer:' would match "computers" but not computation. The rationale for using special symbols in some of the above retrieval modes is that they are special cases and would not be used by the ordinary user.

Other kinds of automatic interpretive matching techniques may be desirable. For example, thesaurus-found related terms, statistical clustering techniques, and special techniques for special data elements like author names (matching certain initials instead of full names, phonetic match of last names, etc.). Linguistic devices analogous to those described above would be needed to control these functions.

Proposed command and response functions and related language features relevant to displaying the Master Index and Thesaurus were discussed in detail in our previous report.[19] The command language is updated here to fit in with the newly developed considerations:

show type vocab data [n lines] term

where

term stands for that word, phrase, or string to be looked up

type stands for the type(s) of relations to be displayed:

'index' -- terms that surround term alphabetically

'phrase' -- terms having word stems in common with term

'thesararus' -- thesaurus relations for term

'relations' -- all of the above relations

vocab specifies the vocabulary(ies) that the related terms must come from; e.g., Mesh, NASA thesaurus, etc.

data specifies the particular data base(s) to be considered; e.g., MEDLINE, NTIS, etc.

n lines specifies the number of lines to be displayed.

The default condition for type and vocab is all.

The related terms displayed by this command will be tagged by short identifiers that may be used to refer to those terms in FIND or SHOW type commands.

A user may wish to specify the type of relation more specifically as, for example, synonyms or narrower (more specific) terms. Sub arguments to the 'thesaurus' argument could be used to make these specifications, as

show thesaurus synomyms term

A user may also wish to extend the relationships found to more than one level in a single command. Thus, to see the terms that are specific to a given term and the terms that are specific to those terms, one could request

show thesaurus specific 2 levels term

The user should also be able to specify that certain thesaurus relations be automatically taken into account in searching: i.e., included in an augmented union set for each term. For example,

find thesaurus specific all (levels) term

which is equivalent to the MEDLINE EXPLODE command.

Conversely, to suppress an automatic use of relationships the user could insert a 'no' argument qualifier; for example:

find no synonyms term

4.3.5.4 Results: Naming, Combining, and Re-searching

The result of a search is a set, or list, of documents. These sets are automatically given names of the form 'set j', where j is a number assigned sequentially. Alternate forms by which to refer to these sets should be 'setj' (no space) and, where ambiguity can be avoided, just the number j. The fuller form including the word "set" is felt to be more descriptive for the inexperienced user and offers less confusion with numbers used in other ways. Of course, the user can always rename the sets to suit his purposes. A convenient way to do this for the current set is with the command 'name set' or just 'name'. Thus

find computer networks name cnet

is equivalent to

find computer networks; rename set k cnet

where 'set k' is the current set name.

Intermediate results may include counts of numbers of documents found under individual terms and partial combinations of terms. To see these counts the command

show count

is employed. If the command were included within the FIND command, the counts would be shwon as they were found; if the 'show count' command came separately, the partial results would be shown after the final result. If the final result is null, it has been found effective[16] to provide the intermediate results automatically to the user; that could be overriden with the argument string: 'no count'. The intermediate sets themselves should be kept at least until the next retrieval operation so that a user can make any of these a recognized, named set without having to reproduce them.

Regularly, the final results from a search as shown to the user would include (1) a restatement of the search query (showing automatic stemming and phrase decomposition, if performed); (2) the count of the number of documents found; and (3) the name given to the newly found set. Internally the system should store the above information for each search together with additional information such as synonymous set names; the actual list of references retrieved; and, at least implicitly, the system(s) and data base(s) searched, the date and time searched, and the identity of the (human) searcher. This information would be available for user review by the command:

show sets [mode] [set i] [set j] ...

where mode would specify if more of less information than the 3 items first listed above were desired; the particular (range of) sets to be reviewed, if other than a full listing were desired, could be specified by the other arguments. A good synonym for 'show sets' might be 'review'.

The user may want to delete some of the sets he has made either because they are too costly in storage (a system may actually limit the number permissible for this reason) or because they are cluttering

up his "thinking space". A DELETE command would accomplish this:

delete set i set j ...

To delete all sets:

delete all sets

To delete a certain number (or all sets numbered before (less than) a given set:

delete number direction set i

where number is a given number or 'all'; direction is either 'before' or 'after'. To renumber the sets in the same order but "closing up the ranks" for the deleted ones, the command would be:

rename sets

For this command the synonyms for the set names would, of course, be transferred with the set to the new set number.

Retrieved search sets may be combined using a COMBINE command with Boolean operators and set names in a way analogous to the use of these operators on terms to be searched in the FIND command. Thus, e.g.,

combine (set 5 or set 6) and set 2 and not set 7

creates a new set with the specified relationship to previous sets. (Note that the parentheses in the example are not necessary since the same left-to-right precedence would have been followed without them, in this case.) Again, we recommend instructing users with an explicit COMBINE command name; expressing the combination function without a command name should be an option for more experienced users. When sets are components of other sets there is a question of how many levels to unravel this structure in the REVIEW command.

The user should be able to intermix searching terms and combining sets. Thus, e.g.;

find energy costs and not set 4

should be allowed. Extending this idea slightly, we see that COMBINE command would not be needed at all; thus

find set 5 or set 6

(Note that this implies the system can distinguish set names from search terms.) Boolean operators should also be implicit FIND commands with the current retrieved set understood as the starting point; thus

and year greater than 1970

should be interpretable as meaning

find set k and year greater than 1970

where 'set k' is the current retrieved set. To make any set the currently active retrieval set the RESTORE command can be used:

restore set 2

The command name 'restore' should not be required; thus, the above would be obtained also by

set 2

If a retrieval or combination function results in a null set, the last previous (final) retrieved set would remain the current set.

It may be desirable to re-run a search statement after a data base update or in an entirely different data base. Adding the argument 'research' (immediately) before a set name would signify that the search statement was to be re-run rather than use the set itself. For example,

find research set 4 and not set 4.

would perform the search originally performed to get set 4 in this new context and then drop out those documents that were in the original set. Since this is likely to be such a useful function it is worth a separate command: 'update set i'.

Normally, a search statement is executed right away - that is, as quickly as the time-sharing system gets around to it. However, in

an optimized system the user should be able to get a delayed execution for lower cost. The user should be able to set up a sequence of statements to be run in this background mode. An important related retrieval function is SDI; that is, the running of a search automatically at each data base update. We shall not discuss further here the various linguistic requirements for specifying the setting up and running of a program and obtaining the results.

4.3.6 Output and Related Functions

The output function refers to the printing or displaying of information from the catalog records -- or full text, if available -- of documents in the data bases. The specifications that may be necessary for the output function include:

(1) What information (data elements, etc.) to be output

(2) For what document set

(3) For what documents in the sets

(4) Where information is to be output

(5) When information is to be output

(6) What format for output

(7) What sort order

The basic ouput command is 'show'. No arguments are required since all the specifications have default conditions. To specify other than the default conditions arguments are required, as described below. In general, the ordering of the arguments is immaterial, except as noted.

The data elements desired are indicated by a string of arguments; e.g.,

show title author abstract

It is often desirable to express a grouping of elements by a single term. For example, 'all' for all elements (the whole catalog record)

and 'citation' for those elements providing the minimum reference information (see discussion in Section 4.1 -- we would reserve the term 'references' to mean bibliographic references in the given document to other documents and 'citing element' to mean a data element from a document that cites the given document.) The 'citation' group is probably best as the default set of elements.

Other kinds of information besides data elements, as such, may be called for if the system has the capability. Thus,

show text

could call for a display of the full text while

show match

could call for output showing why each document was matched -- e.g., by "highlighting" those words in title or abstract that match the search statement.

To specify which set one wants to output the name of that set is used as an argument:

show set 5

In the default case the current set is assumed. Also, in the default case it is assumed that all documents in the set are wanted. To select a subset of the documents the argument 'documents' (abbreviation 'docs' or 'doc') is used:

show abstract documents 3 7 to 10 15

gets the abstract for the third, seventh through tenth, and fifteenth documents in the current set. The connector operator 'to' could be replaced by a hyphen.

The 'documents' argument can be used with the FIND command to generate a new set that is a subset of the current. Thus,

find docs 7-10

will create a new set with 4 documents from the current set.

It is assumed that a user can interrupt the output at any time. If the system does not permit interrupting, or if the system wants to avoid an excessive amount of online output, it may stop the output and ask the user if he wants to see more. The command

show more

would be a positive reply. The arguments 'from' and 'after' would also be used to indicate that all documents after a certain number were wanted; e.g.,

show from doc 7

A user on seeing a title for a given document in a string of document titles might want to see more information on that document. The user could do this by interrupting and then issuing the following commands:

show abstract document 7
show title from doc 8

To add document 7 to a special saved set before continuing the user could issue these commands:

find doc 7; or saveset; rename set k+1 as saveset;
show set k title from doc 8

where 'set k' is the current set. In order not to create the superflouous set k+1, a command 'keep' might be defined; e.g.,

keep docs 5 8-10 (in) saveset

would add 4 documents to set saveset without creating any new set names. If 'saveset' were not names, a systems-defined, default set would be assumed.

To specify a different document order than the one provided by the system -- usually an (approximate) inverse chronological order -- the ORDER argument is used:

show order field mode

where field specifies the element or group of elements to sort on:

mode specifies the mode of sort; e.g., forward or reverse.

The output format would depend in part on the SPEAK mode: VERBOSE being more explanatory about what the element fields are. Other formats would be specified by other arguments to the SHOW command.

The default situation has the output going to the user at the terminal. To send output to be printed offline the argument 'offline' would be used in the SHOW command. The address to which offline output should be sent should be stored by the system; getting the information about the different parts of the address appears to be one situation in which the prompting mode has advantages. As with the FIND command, there may be various levels, besides offline, of delay in the execution of the SHOW command.

4.3:7 Instruction and Status Review

We have previously discussed the 'help' and 'explain concept' commands and some other facets of the instructional features of the retrieval language (see, especially Sections 2.1, 3.3 and 4.1). Some additional features desirable to enhance instruction are discussed in this section.

The command 'explain' without any arguments can be taken to request an explanation of the last message or current content. The command

explain message

can be taken to request explanation of a given message or message component identified by the argument message which could be a prefix of, or a tag associated with, the message.

The user should be able to "turn off" lengthy instructional messages once he has seen them enough to learn their message. The command

speak message terse

would request this. The command

speak message verbose

would reverse the setting and 'explain' or 'explain verbose' or 'explain message verbose' or 'explain more' would give the fuller explanation at the current time without actually resetting to the VERBOSE mode.

According to the simple-basic-core principle the user should be shown only a few basic features to start with. However, the system should occasionally prompt the user to try additional features. To do this effectively the system should keep track of what features the user has employed and explain, in appropriate contexts, additional features that might prove useful. This dynamic instruction would be guided, as the whole interaction is, by user mode settings and specific requests.

Online human instruction would be valuable at times, although perhaps costly. To invoke such help one might use the command

help human

after which a free-form dialog between instructor and user could ensue in which the execution of regular commands would be suspended until the regular mode were reinstated.

More generally, the user might want to communicate with other persons via the computer. A message-sending command might have the following syntax:

send mode to name address message message

where name and address tell where to send the message

mode expresses a mode of transmission -- e.g., immediate or offline

message is the message itself -- which could come from a file rather than from the command line

Status information would be provided to the user as part of the regular dialog and in response to certain EXPLAIN and SHOW command options. Many kinds of status informaton have already been discussed.

Some specific status that can be, or should be, available, may be listed:

(1) systems potentially and currently available;

(2) date bases potentially and currently available;

(3) dialog modes currently set;

(4) cumulative and incremental time and cost considerations.

4.3.8 Saving, Sharing and Reviewing Results

One area that current retrieval systems are just beginning to develop is the saving, reusing, and sharing of search results from one session to another. To save a retrieved set a saved file may be opened:

open file

where file is the name of a previously created or new file. Sets may then be saved in this file with the SAVE command:

save set j set k

The information about the sets mentioned in Section 4.3.5.5 should be kept in the saved file. In fact, sometimes it may be desired only to save the search statements. These sets may then be used in subsequent sessions by their creator or someone else with the creator's permission. As a further aid in recording and communicating results the saved sets and files should be annotatable by the users.

To distinguish two sets that may have been given the same name it may be necessary to prefix their given names with some of the status information. To restore saved sets or files the RESTORE command may be used:

restore file [set i] [set k]

where all or, optionally, some of the sets in file are added to the current file from which they may be used in the same ways as sets created during the current session.

Two other kinds of saved files may be useful. Storing the interactive dialog in a monitor file can be useful for systems analysis and

as a means by which the user, especially a user at a display terminal, can "page back" to see previous dialog. It may also be useful to store the output from various searches in a common file as is now done in CONIT. The monitor file would be automatically updated, if used at all. The output saved file would be opened as the other saved files and would be updated by the use of the argument 'save' in the SHOW command. A VIEW command would be used to display from these files; e.g.:

view [file] page n

where the previously used file would be assumed if not expressed. Other examples:

view certain [page]

where certain = last, next, previous, first, etc.

4.4 Summary

We have described some specific plans for a common retrieval language based on certain principles of user/system interaction and desired features of interactive languages. Having examined the problems of using unrestricted English as a common retrieval language, we have tried to determine the general and particular features of a language that would be simple, easy-to-use, extensible and containing at least some of the elements of English that appear helpful for interactive dialog. The language is intended to have an "open" format and make use of the best features of existing languages.

The language as described is neither final nor complete in that it must be tested and many additional functions may be required. We have tried, however, to (1) suggest the variety of functions desirable in a retrieval system (2) raise issues with respect to the linguistic features to express those functions; and (3) suggest some particular answers to these issues. It is noted that answers depend as much on what set of functions is decided on in any given implementation as on

linguistic principles, as such. How the language questions are related to questions of interfaces and networking is discussed in following sections.

5. MESSAGE INTERPRETATION AND PROTOCOLS IN AN INTERFACE

Our experience with the design, implementation, and evaluation of the experimental interface, CONIT, has led us to a clearer understanding of what functions need to be performed by a translating interface in a computer network situation. In particular, we have been led to consider the character of the timing and translation of messages among the interacting but independent and heterogeneous processes involved in the interface operations. One special character of this interchange derives from the fact that although the messages coming from the retrieval systems were designed for human interpretation, in this situation they are actually interpreted by a computer process: the interface.

We hope that our characterizations may lead to the development of a model that will be useful for aiding in the resolution of three kinds of problems in the area of networked interfaces. The first problem is an adequate general characterization of message handling functions, timing, and translations for networked interfaces. The second problem is the design of mechanism for conveniently describing the actual messages to be transmitted from a specified common interface to particular retrieval systems in response to specified conditions and messages from the given systems and from a user. The third problem is the design of a software structure which provides an effective and flexible mechanism for carrying out some major part of the interface functions as specified by some mechanism such as one associated with problem area two mentioned above. We shall discuss the utility of the model for addressing these problems after describing the model.

5.1 Simple Model

We shall first describe our initial formulations of these problems and their shortcomings. For most retrieval systems -- as for most computer systems that work in an interactive, time-sharing mode with human users -- the usually accepted basic mode of operation is one in

which each party in the dialog -- the human user and the computer system -- takes turns in sending messages to the other. Thus, typically, the user first makes a request of the system; then the system interprets and responds to the user with some message of its own. This cycle of non-overlapping, sequential messages* is repeated after the user, having waited for the conclusion of the message from the system, digests that message and decides on his next course of action -- which is expressed as a second message to the system. This sequential mode is illustrated diagramatically in Fig. 6.

The extension of this simple sequential mode of operation to the interface situation is diagrammed in Fig. 7. In this mode the user in each cycle first sends a message (M1) to the interface; then the interface interprets this message and translates it into a request to the retrieval system (M2); next, the retrieval system sends its response (M3) to the interface which, finally, translates it into a response (M4) to the user's original request.

Three modifications to this simple 4-step cycle may be enumerated which will make for a more realistic model of the necessary interface operations. In the first place, the interface might well respond directly to a user request -- say, a request asking what systems are accessible from the interface -- without need to go to any retrieval system; thus messages M2 and M3 would be short circuited in this case by action purely <u>local</u> to the interface. Secondly, any such message M4 from interface to user might be interrupted by the user sending an "interrupt" or "break message", the interrupt would occur during M4 and cause the interface to stop sending M4 immediately and to return

*Typically, a user issues a "command" and the system returns with a "response" message. However, the system can also "command" a response from the user, who may also wish simply to send an informative message (e.g., gripe) to the system. The general term "message" will be used to cover all these situations; different message types will be indicated as needed.

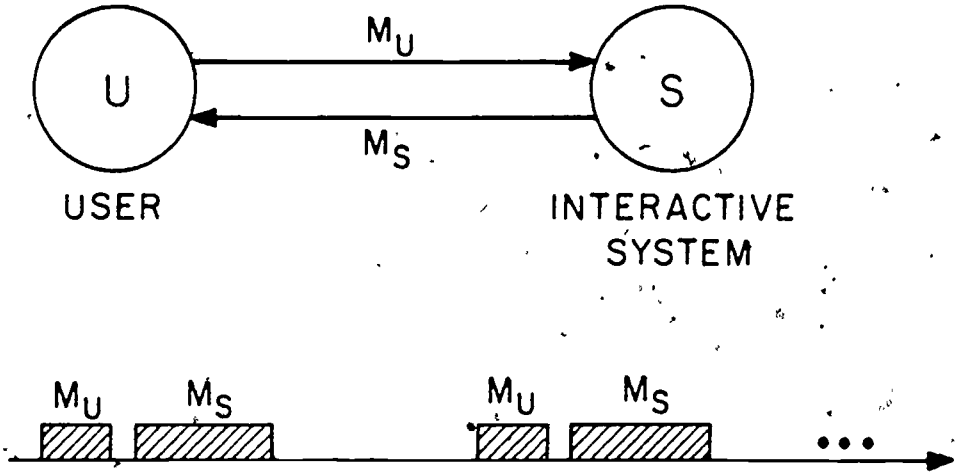


FIGURE 6 TIME DIAGRAM OF USER/SYSTEM MESSAGE FLOW FOR SIMPLE SEQUENTIAL OPERATION

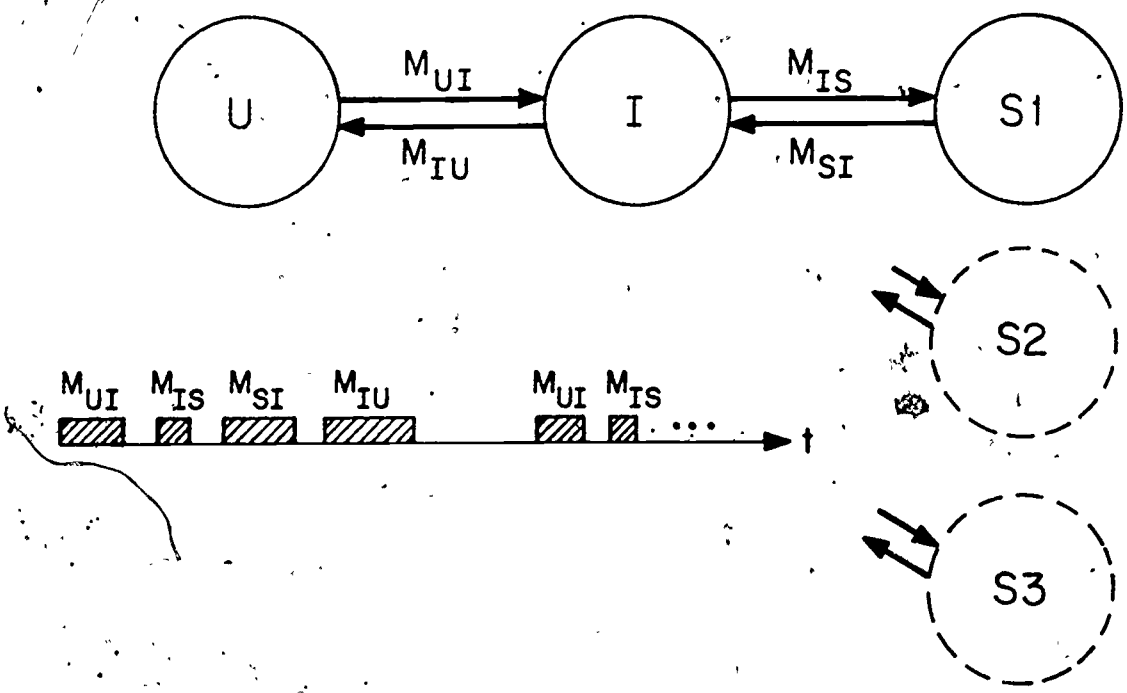


FIGURE 7. TIME DIAGRAM OF MESSAGE FLOW WITH INTERFACE PROCESS FOR SIMPLE SEQUENTIAL OPERATION

to a state awaiting further user requests. Thirdly, one type of user command message would be to select a different retrieval system for searching, thus several retrieval systems could appear sequentially, on different cycles, as the recipient and transmitter, respectively, of messages M2 and M3.

This modified sequential model corresponds generally to the basic structure of our early experimental interface, CONIT. Also, as we have indicated in our description of CONIT in Section 2, a simple translation scheme was implemented in which a pair of translation tables was devised to effect the translations for each retrieval system: one table to translate user input to retrieval system input (message M1 to M2) and a second table to translate retrieval system output to user (message M3 to M4). This translation is a straightforward conversion of specified strings from the input character stream to similarly fixed and prespecified output strings.

5.2 Limitations of Simple Model

We knew at the outset, of course, that these sequential operations and simple translations would not suffice for everything we might wish the interface to do; the degree to which they were effective and the particular ways in which it turned out they were insufficient provide a valuable basis for analyzing the complexities of the interface situation. Some of these complexities are discussed next.

5.2.1 Interface/Systems Dialog Unmediated by User

A single user request may require a series of interactions between the interface and a remote system rather than the single pair of messages M2 and M3 implied by the simple model. An important example of this occurs when the user requests the selection of a new system through the PICK command. Here the interface must go through an extended exchange of messages with the retrieval system. Even within the limits of a single retrieval system a series of messages may be required. For example, an output request by a user which selects a

discontinuous subset of documents to be output (e.g., SHOW DOCUMENTS 1 4 7-9) may require a series of output requests be sent to a system that cannot handle such a request in one command.

In some instances it may be desirable for the interface to initiate an interaction with the retrieval system without any explicit user request. For example, a retrieval system may drop a user who does not react with the system for more than a given amount of time -- say 15 minutes. It is desirable for the interface to keep track of status information like the time since the last interaction. The interface could then send a simple request (e.g., asking for the time used in current session) to forestall the line dropping while checking the status of the connection to the retrieval system.

5.2.2 Indefinite Nature of Systems Response

The general nature of, and particular realization of, system messages may be difficult to predict for a variety of reasons as outlined below:

(1) In general, it may be difficult to know the precise nature of the responses to be expected from the retrieval systems. Retrieval-system designers devise the response repertoire of their system to be largely self-explanatory to a human user. To the extent that they are successful -- or believe so -- they may not feel the compulsion to fully describe these responses in any written documentation like, for example, a user's manual. While the common message types may be fairly easy to uncover, messages for special situations (e.g., error conditions) may be very difficult to learn about through the standard inquiry channels of (1) written documentation, (2) informal communication with system designers or users; and (3) experimentation with the system itself.

To compound these problems the retrieval systems are often very dynamic in their construction -- especially in regard to the system-to-user dialog. It is not unusual for any given system to experience several changes of this kind in the course of a month -- often with no

prior warning, or only a very general notice perhaps to the effect that a "new system" is "about to appear." A change in the logoff message, for example, may seem innocent enough and be easily understood by a human user but could cause serious problems for simple-minded computer algorithm that was looking for one fixed string -- say, "LOGOFF" -- and finds another -- say, "system disconnected."

(2) In particular, it may be difficult to know when a message has been completed. Usually there is a "user prompt" which is a particular string of characters that signifies that the message from the system is completed and the system is prepared for a new message from the user. However, sometimes a system may depart from this scheme, for example, when it asks the user to respond to a particular question -- say, "Do you want to continue printing output?"

The difficulty of knowing when a message is completed is compounded by the stochastic nature of the messages: because of the inherent character of time-sharing systems, messages may start being transmitted at some indeterminate time, may be interrupted temporarily for another unknown interval, and be concluded at a time of similar indefiniteness. The interface must wait a reasonable amount of time before concluding that no further message is coming from the system but it must not keep the user waiting an unreasonable amount of time either -- see discussion on responsiveness in Section 3. The appropriate timing of time-out signals for the interface and what message to the user and other functions should be performed at these times are clearly important issues.

(3) Variable Messages. Most messages, or crucial parts thereof, are variable in content by their very nature. Messages of this type include: output about documents; the message telling how many, if any, documents were found in a search; and news given at login or in response to an explicit request.

5.2.3 Unexpected or Unpredictable Messages. Communication channels can generate erroneous transmissions. Moreover, computer systems can and do get sick and die at unpredictable times. The messages received from system channels at such times can vary from (1) nothing (there may be a simple line dropout with or without line-disconnect

notice) to (2) slightly distorted messages to (3) gibberish to (4) "wrong" messages (as when responding to transmission-caused "wrong" command to (5) a message stating the time of initiation and expected duration of an outage. These latter messages may be of a well-specified form or may be completely free form. At such occasions the control of message response from system channels may change. For example, control may shift from a retrieval system to a time-sharing supervisor (e.g., IBM TSO) or to an intermediate network through which connection to the retrieval system was arranged (e.g., TYMSHARE or ARPA network). Such changes of control can dictate corresponding changes in message form (e.g, end-of-message indication) and message content (e.g, a line dropout indication as opposed to the expected response to a previous command). The interface must be alert for these possibilities, try to diagnose them correctly, and be prepared to act appropriately:

5.2.4 Overlapping of Messages. Contrary to the assumption of strict sequentiality in messages made in the simple model, there is need to consider a high potential for overlapping messages beyond just user interrupts. Because of the variable nature of system responses in terms of timing, length, and content, it is important to consider taking advantage of the full-duplex potential of the communication channels. For example, it is necessary to be prepared to accept and react to an unexpected message of the type mentioned in Section 5.2.3, above, which could occur while the interface is sending a message to the system or is interacting with the user. Furthermore, there is the possibility of much greater efficiency and responsiveness to the user if the interface is capable of interacting with the user while it is also doing so with a retrieval system, especially where long interactions are involved.

For example, the interface should keep the user informed, during the long connection process of success or failure -- or, especially, that intermediate situation that frequently creates uncertainty and anxiety in the user: delay. (See Boies [24] for discussion of how "time uncertainty" adversely affects users.) Also, for efficiency and to avoid

delay, the user should be given the initial parts of responses from the retrieval systems, as for document output or news messages, while those responses are still being received at the interface.

5.2.5 Multiple Simultaneous Retrieval Systems

It may be desirable to search several retrieval systems at the same time or, at least, alternately and in such close proximity that it would be inefficient to login and logout for each search. The ultimate interface system would provide for the simultaneous searching of multiple data bases wherever they may exist so as to allow for greater responsiveness and comprehensiveness of retrieval function for the user.

5.3 Towards A More Comprehensive Characterization

The limitations of the simple model described above in Section 5.2 led us to consider what elements would be required in a more comprehensive and adequate characterization of message communication in a networked interface. This section includes the beginnings of such a more comprehensive characterization.

It should be remembered that the interface we are considering connects a user to existing, independent retrieval systems without requiring any change in these systems. If standardized network retrieval protocols were devised, and if retrieval systems were modified to adhere to these standards, many of the problems we have been describing could be circumvented or, at least, handled in a fairly straightforward way as we shall discuss in Section 5.4. However, it is well to consider the complexitires as they now exist because (1) in so doing we may help point the way toward and encourage standards and (2) we may never, or not for a long time, achieve the needed standards.

5.3.1 Communicants and Communications

The kind of network we are investigating is characterized by communicants sending each other messages. A message is generally either (1) an imperative -- i.e., a request for some action expressed as a command -- or (2) a response to some imperative. However, an unrequested

declarative -- e.g., "The systems will be going down in 5 minutes" -- or other mixed types are possible. Even a declarative is often an implied kind of imperative, e.g., for the previous example: "Please finish up and log off in 5 minutes or your session will be terminated (abruptly) by the system".

The communicants for the interface situation are (1) the interface itself, (2) human users, and (3) computer-based retrieval systems, and occasionally (4), other computer systems like operating systems for individual computers or network communication processes whose function in the retrieval application is to establish and maintain the connection to the retrieval system. We are, in general, interested only in those types of messages that would be generated by, or intended for, the human user in the course of the retrieval application. We are not, for our present purposes, concerned with the lower-level, inter-process and inter-system protocols upon which the higher-level, human-oriented message flow takes place. Thus, we are not concerned with that "communications subnetwork" of minicomputer processors that provide the inter-computer communications nor with the protocols among these communication processors or between the communication processors and the host computers on the network in so far as all these protocols are essentially transparent to the retrieval systems and human users.

5.3.2 Communicants as Rule-Governed Processes

The communicants can be viewed as processes which generate, interpret, and respond to messages. We would like to characterize the rules by which this interpretation is (or could be or should be) done. One kind of rule has to do with the time during which communication will be accepted. A second kind of rule concerns the protocols for a message; what format it must have, what signifies that it is completed, etc. A third kind of rule concerns the actual rules for interpretation and response to particular messages. The most comprehensive level of concern with respect to rule execution has to do with the data, both data internal to the communicating process and external events, upon which the rules are

applied in order to determine the particular response messages.

All these kinds of rules are, for the interface itself, open to the determinations of the interface system designer and may be optimized by him with respect to his chosen parameters only under the constraint that the other processes are suitably respected. With respect to the retrieval systems, the rules are largely fixed and, under the guidelines of our approach, not under interface control. The one major exception to this lack of control is that most systems will have two or more modes of operation in which the output messages -- and, possibly, the input commands -- may take different forms: for example, a short form for experienced users and a longer instructional form for inexperienced users. The interface can set this mode and, in general, would choose the more compact form for efficiency.

Knowledge at the interface of rules at the retrieval systems varies with the type of rule. Knowledge of timing and format rules generally can be well established. Rules of interpretation and response can be known in general terms subject to the limitations mentioned in Section 5.2. Actual responses cannot, in general, be predetermined since they often depend on the detailed contents of the data bases. Except for interaction with the index files in an implementation of our Master Index and Thesaurus concept, responses to messages involving interaction with the data bases can only be known a posteriori by observing actual responses.

Knowledge of the human user as a rule-obeying communicant is much less well defined. As an input device the human can accept a wide range of timing and format although, depending on the user, some formats are likely to be more effective than others. As an output device the user is forced to accept the format that the interface demands; i.e., the common command language. As a message interpreter and responder the individual human is largely an enigma, although studies [18,21-24,35-37] have shed light on the nature of typical users. However, the interface can strongly influence the nature of the response through instructions, suggestions, and particular queries to the user.

5.3.3 Structure and Timing Considerations

The network structure has the interface itself as a mediator between a user and several interactive information systems. Thus, a diagram of our extended model, shown in Fig. 8, looks structurally similar to that of the Simple Model of Fig. 7 with the main difference being the explicit recognition of multiple, simultaneously-connected information systems. It is also recognized that any connected "information system" is not necessarily a single, monolithic system but can appear to the interface at various stages of the dynamic networking process as a network connector or a host-computer operating system. Generally, when the interface has established connection to the retrieval system these intermediate stages become transparent and can be ignored until a disconnect -- either intended or accidential -- causes them to reassert themselves.

It is worthwhile, parenthetically, to consider the question of multiple, simultaneous users. This multiplexed situation clearly would be part of any efficient operational interface-form of networking. However, it is quite conceivable that the multiplexing needed to handle multiple users can be accomplished entirely -- or, at least in large measure -- by the systems and networks in which the interface resides or to which it is connected to. In any case, the issue of multiple users is a separable one.

An important generalization to the simple interface model is in the area of message timing. As was pointed out in the previous section (5.2), we want to be able to consider a considerable amount of overlapping in time among messages. Basically, messages from either user or any system are conceived as arriving at the interface at some later time. Conversely, while messages are being received, the interface may be sending messages to any combination of systems and user.

However, in the retrieval application the timing of the reaction to messages is usually not too critical. In particular, the interface can generally wait a minute or more to respond and still not cause any

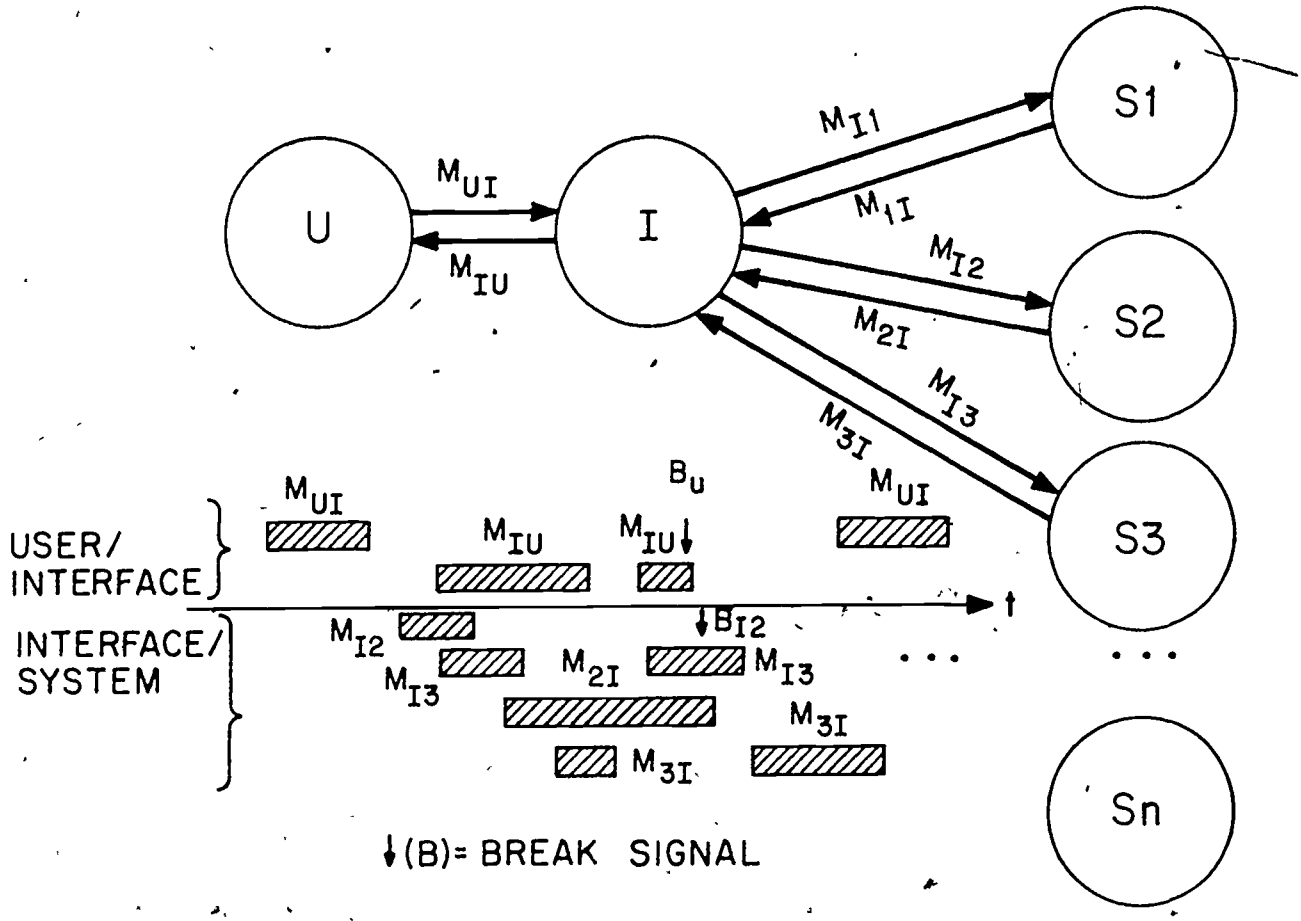


FIGURE 8 TIME DIAGRAM OF TYPICAL MESSAGE FLOW FOR GENERAL INTERFACE SITUATION

problems. In general this means a message can be interpreted and responded to before considering any other messages that may have arrived after the arrival of the given message. The most critical timing is in the login phase because timeouts may occur if responses are not sent to some messages within a period of the order of a minute. In order to avoid such timeouts the interface can be programmed to follow through with the login to one system before starting another login or reacting to a message from the user or another system.

Occasionally, it may be desirable to hold up the processing of one message until an incoming message is completed: for example, a user command to stop waiting for a response from a retrieval system if that response is just starting to arrive.

The fact that a message is initially interpreted does not necessarily mean that the full response to it is given at that time. User interupts, for example, may simply be noted for action at a later time, perhaps, when an ongoing operation is completed. This situation can be discussed further after we describe in greater detail in the following section that nature of the rules to be followed by the interface.

5.3.4 Message-Handling Rules for the Interface

The rules for interpreting and responding to messages at the interface can be thought of as operating on input message streams and generating output, or response, messages. One generalization over the simple model is that response messages may be directed to more than one communicant as the result of a single rule -- typically, say, to the user and the currently active retrieval system.

Another major sophistication for the rules is that they be context sensitive through the mechanism of state variables. Thus, in addition to finding a particular match in the input stream, a rule would require that certain state variables have specified values before the rule would be executed. A rule could also include the setting of given values for state variables in its execution. A state variable may specify a very general state: for example, that the user is using VERBOSE mode; or it may indicate a very specific situation: for example,

that the interface has just sent the password in the login procedure to system X and is awaiting the response. Thus the rules can be set up to relate to, and "step through" a sequence of very specific situations for various combinations of general modes in effect.

The rule must identify some part of the input stream as meeting a particular criterion for match in order that the rule be invoked -- assuming of course that the state variables also match, as just described. That part of the rule that specifies the nature of the match may be called the rule match, or simply match. Also, there is a pointer which identifies that point (i.e, character) in the input stream at which the interface begins a scan of the stream to ascertain whether any rule match is satisfied -- scanning going in the positive direction i.e., the direction in which characters have been added. A rule would include the specification of how to increment the pointer.

Normally, after a rule is executed, a search is begun for the next matching rule in accordance with the rules of message priority as, for example, indicated above in Section 5.3.3 and rule ordering as discussed below in Section 5.3.6. However, it is conceivable that control should be otherwise directed after a given rule; the capability to provide this kind of direction should also be expressible in the rule. Fig. 9 schematizes the kind of structure we have in mind.

5.3.5 Message Formats, Timing and Segments

Now we describe in greater detail the actual rule-matching and message-generation operations required in the interface. First, we need to consider the format of the incoming messages. These messages can be decomposed into segments; the most common and natural segment is a line; i.e., the character string ended by a new line or other line-ending character, like carriage return or line feed. For some systems, and in certain situations, only a partial line will be sent. This will happen, typically, where a system has a user prompt that does not end in an end-of-line type character: as, for example, just a question mark on a line.

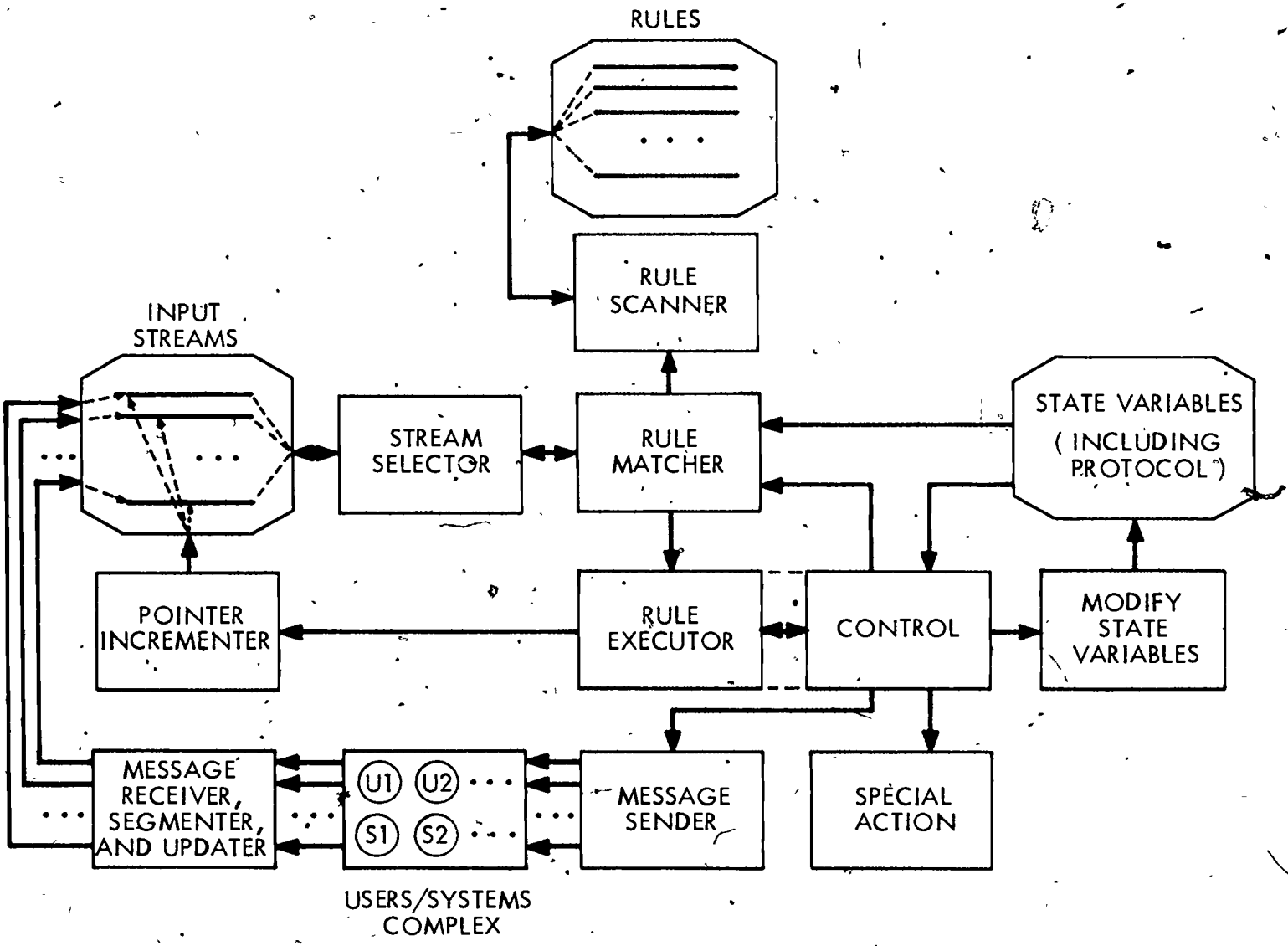


FIGURE 9 MODEL OF MESSAGE INTERPRETATION AND RESPONSE COMPONENTS IN INTERFACE

The basic mode of operation would then be to add a message segment to the input buffer when it is received and to perform rule matching on the incremented input stream which, in general, would represent a partial message. Of course, a completed message would be a special case of a partial message and particular rule matches would attempt to identify end-of-message segments for special attention. The set of delimiters specifying message segment boundaries would be dynamically set by the rules. The size of these sets is quite small for most common systems; it usually ranges from just the end-of-line characters to that minimal set augmented by one or two punctuation characters like question mark or colon.

A timing function should be built into the message segment handling operation such that characters coming in immediately after (i.e., at a time interval no greater than that determined by the BAUD rate) a non end-of-line type character delimiter are appended to the message segment. This would avoid forcing the rules to try to handle partial lines where end-of-segment delimiters are innocently included within regular lines without having an end-of-segment function. Conversely, if there were a segment not ending in one of the currently recognized delimiters, a timeout function would come into effect to force the transmission of this (unexpectedly) short segment into the input stream as well as setting a state variable to identify this condition. Two kinds of situations could induce this kind of timeout: (1) the rules simply had not properly specified the current delimiter set or (2) due to error conditions or the stochastic timing idiosyncrasies of the time-sharing mode of operation, the end of a segment had been inordinately delayed. Note that a null segment would be a special case of this latter situation and would be identified by another particular state variable.

If it were desired to base a rule match on some features of a partial message that overran segment boundaries, this could be accomplished by proper setting of state variables, the input stream pointer, and/or the rule match. Interrupt messages as well as timeouts should

set state variables so as to allow for all rules to be expressed in a uniform input/state description of context.

5.3.6 Rule Matching Criteria, Transformations, and Ordering

Rules may be of certain rule types based on the kind of matching functions in effect for the rule match. For example, it may be desirable to ignore upper-case/lower-case distinctions for alphabetic characters. Other rule types will be discussed below after a more comprehensive discussion of the general matching criteria is accomplished.

It is clear that rule matches should be capable of specifying any given character or fixed character string, whether these characters be alphabetic, punctuation, non-printing characters or, in general, any code. In addition it is advantageous to be able to have variable features of the input stream be specified in the rule match. For example, it may be recognized that a character string (of indefinite length) that appears after a user FIND command is to be taken as a (free vocabulary) expression of a search topic and should be placed in a certain position in the output message. A symbology is needed to represent such a variable string for both the rule match and the rule message.

Another kind of variable element would stand for some class of characters sav: end-of-line, alphabetic, non-alphabetic, numeric, command delimiter (e.g., semicolon or end-of-line), etc. This kind of variable combined with the variable-length element would provide the means to specify variable words and phrases of a given character; for example, a number would be a variable-length string of numeric characters.

It is desirable to be able to specify that some identifiable elements of the input stream undergo some particular functional transformation, that is not (easily) expressible by the string manipulations of the rules themselves, before being deposited in a state variable or output message. For example, an arithmetic function may need to be performed on the number n, represented by a given string in the input --

which in turn may represent, say, the number of the first document to output -- in order to properly translate to the appropriate command message -- e.g, "PRINT SKIP n-1". The symbology for expressing this kind of transformation needs to be developed for incorporation into the model.

Rules would be ordered and the search for a matching rule would proceed by that order. The first rule matched would be executed. There would always be a default rule, in general, or in a particular context, so that unanticipated or default occurences could be handled. All variable states expressed by a rule would have to be satisfied for the rule to match; a variable state not expressed by the rule would be ignored in the matching operation. Rule matches would generally be ordered from longest to shortest so that rules depending on more precise context would take precedence over those broader or default contexts. The specific ordering of rules in cases where ordering would not ever effect the actual choice of rule for any possible input streams and state variables would depend on such factors as whether the rules were intended primarily for exposition to a human analyst or for actual execution. In the former case, an ordering based on state variables might be preferred; in the latter case, an ordering based on a computer sort order (e.g., alphabetic) might be preferred for efficiency of searching.

The nature of the operations provided for in this model is schematized in Fig. 9.

5.4 Retrieval Protocols in Cooperative Networks

The description above of functions required in a networked interface for interactive retrieval systems, while reasonably comprehensive in coverage of the kinds of functions required, is limited in three respects. First, as was pointed out in the beginning of Section 5.3, we have assumed independent retrieval systems that could not be changed. Second, we have tended to lump all the functions together in one undifferentiated mass with regard to the different levels involved. Third, we have tended to ignore the structure of the network in which the interface would reside. In this section we shall take a very preliminary view of what might result if we could go beyond these limitations.

In the first place, if networked systems could achieve that degree of cooperation such that standardized communication protocols could be agreed upon, then many of the problems of indefinite, unexpected, and unpredictable messages mentioned in Sections 5.2.2. and 5.2.3 could be circumvented or, at least, reduced in scope. Thus, for example, message completions, acknowledgements and system dropouts would all be handled in standard ways.

In the second place, the network structure in which the interface resides has a strong impact on how interface functions should be performed. In particular, we see in such ARPANET efforts as the RSEXEC and the National Software Works [12-14, 38] (see Section 1.2) the development of a distributed-computation approach to resource sharing based on common protocols for intercommunicating among dispersed processes to handle a given application.

Thus, for example, it is suggested that any major application handled in the network, like interfacing to retrieval systems in a virtual mode, be implemented in several separated, but interconnected and cooperating processes. There are at least two main reasons for resource sharing through this kind of structure: reliability and efficiency. Reliability is achieved by having separate processes each of which can individually handle the application. Thus, if one process is unavailable for any reason -- failure in hardware, software, or communications channels -- the user can be routed to another process providing the same functions. Of course, the appropriate switching mechanisms must be available. Greater efficiency through load-sharing for example, can be achieved by routing users to less busy processes, rather than to overloaded ones.

A second aspect to emerging networked structures, where distributed processes are connected to and serve likewise distributed users at terminals, is the recognition of two kinds of processes involved: user processes and server processes. This distinction is in accordance with the actual nature of the network situation: users are attached to individual host computers and are, in general, required to make connection

to serving processes which reside in separate host computers. Thus it is quite natural, for each individual application, to have in each host computer a user process that takes all requests for that application and establishes in an appropriate and uniform way, connections to suitable serving processes in the given and other computers. Uniform access methods are key to effective network operations.

The structure for a networked interface containing distributed user and server processes for retrieval is exemplified by the diagram in Fig. 10, where the overall interface process is still called "CONIT". If we compare Fig.10 with Fig. 1 we see that the module labeled "User Interface" in the figure is included within the user process and the module labeled "Translation" is included within the server process. The function "Interface Management" in the first figure is distributed over both user and server processes in this revised picture.

In this revised picture the communication between user and server is accomplished through agreed upon procedures and formats that may be termed the retrieval protocol. In particular, the user process translates a user request into one in a common request protocol, (labeled Command Protocol in Fig. 1) which the server process translates into the appropriate form for the retrieval program. Correspondingly, in the reverse direction, the server process translates a response from a retrieval program to a common response protocol which is sent to the user process for conversion to a form suitable for presentation to the user. In the most general sense the protocol includes all the procedures by which the user and server processes communicate with each other as well as all the status information for each user, including all the various functions and function responses discussed in Section 4. We note that between user and server -- i.e., intra interface -- we can conventionalize and standardize the protcols and thus avoid many of the problems of networked communications as described in Sections 5.2 and 5.3.

It is worthwhile to consider the functions of the server process in greater detail. For non-cooperating systems of the kind we are

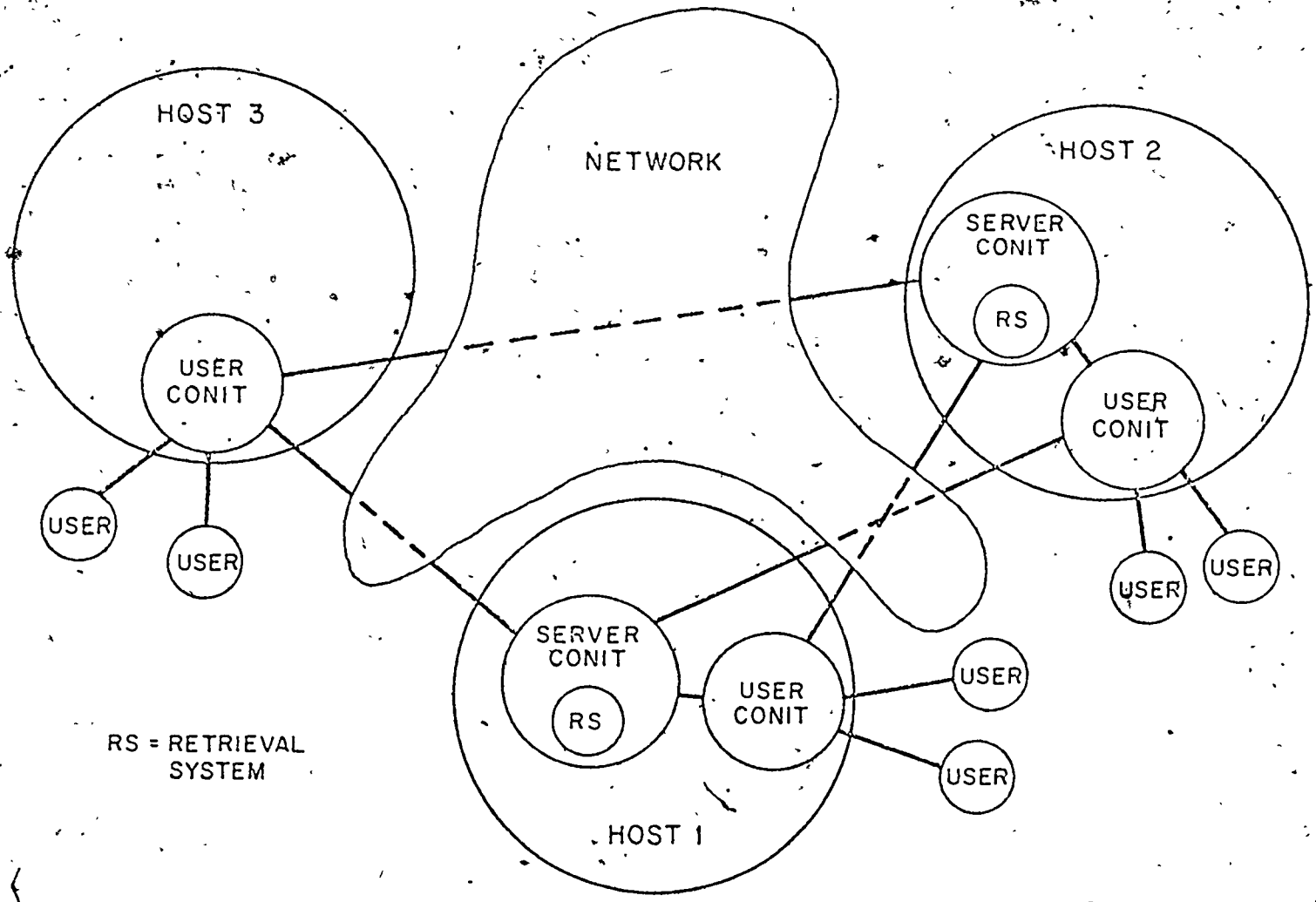
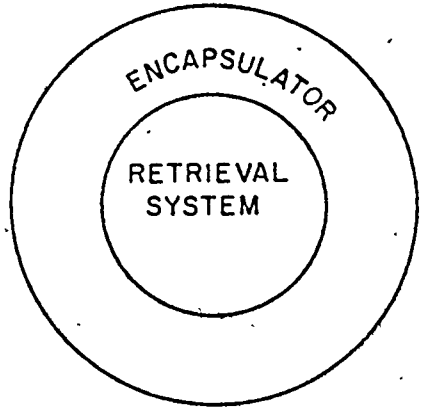


FIGURE 10 DIAGRAM OF RETRIEVAL NETWORK IN WHICH INTERFACE IS DISTRIBUTED IN USER AND SERVER PROGRAMS

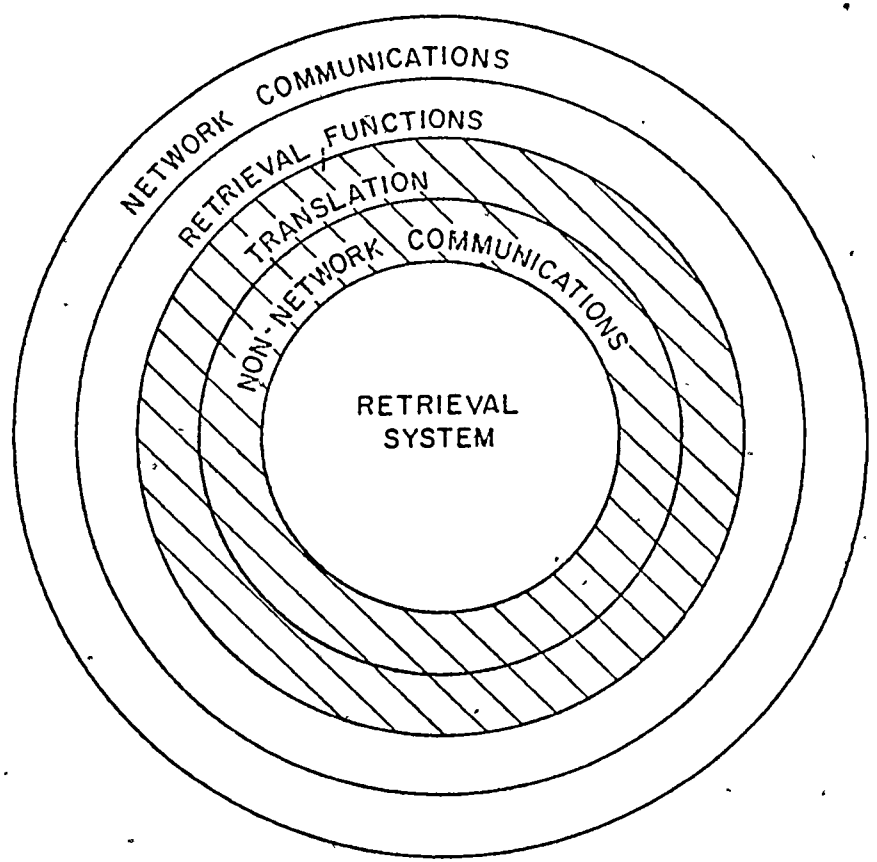
currently dealing with, the server must communicate with the rest of the network with the standard protocols while also handling the individual non-standard features of each retrieval system. We may say that the server "encapsulates" the retrieval systems and makes them appear to the rest of the network as if they performed standard functions according to conventional protocols. The "encapsulator" notion is implicit in Fig. 10 and made explicit in Fig. 11A.

The encapsulation function itself may be subdivided into a number of functions that successively carry and transform the standard protocols into the retrieval systems and, then, the responses of the retrieval system backout to the network, as shown in Fig. 11B. First the encapsulator must handle the establishment and maintenance of connections to the rest of the network. Next, the retrieval protocol must be interpreted and other management (e.g., status keeping) functions must be performed. Third, the interpreted protocol functions must be translated into commands for the retrieval system. Fourth, the translated commands must be passed along to the retrieval systems, possibly through non-network, non-standard communications channels, if the retrieval systems are so situated. Similarly, the retrieval system responses must pass successively through these functional rings back out to the standard network interface maintained by the encapsulator.

One advantage of the user/server process structure is that a new retrieval system that follows the common protocol can be added to the network directly, i.e., no new translation modules are needed. In fact, the third and fourth (shaded) rings in Fig. 11B cah be eliminated in standard network operations. For a retrieval system not following the protocol, at least a well-defined translation procedure is implicitly defined for the encapsulator. In addition, the intra interface protocols, being freed from a requirement for human intelligibility, can be concise and, therefore, more efficient for processing and communications. Also, having such separate communications protocols tends to isolate the surface languages and, therefore, make it easier to change those languages without making major modifications to the basic interface operations.



(a) SERVER (Simplified View)



(b) SERVER (Detailed View)

FIGURE 11 SCHEMATIC DIAGRAMS OF SERVER COMPONENTS

One question for future research is how to characterize the structure of the interface more finely in terms of function so that higher-level semantic functions, e.g., command interpretation, are more clearly separable from lower-level (syntactic) functions, e.g., character string handling. In any case, our current conclusions on the future role of communications and protocols in networked interfaces are given in Sections 6.1 and 6.3 below.

6. EVALUATION

We have described the nature of the retrieval networking problem of coupling heterogeneous independent interactive systems; discussed general approaches to its solution; outlined specific techniques; and described an experimental system to aid in analyzing the problem. In this section we evaluate the general prospects for resolving the problem and the several particular approaches we have considered. At this stage the evaluations are still tentative; more extensive experimentation and analysis will be needed to draw firm conclusions in many areas.

6.1 Physical Interconnections

Rapid developments currently in progress in the field of computer networking should soon alleviate current problems in the physical interconnection of interfaces and retrieval systems. Most of the major operational retrieval systems already are, or soon will be, accessible via national and international computer networks. It should be possible to build the interface components on hosts that are part of, or can be easily attachable to, these networks.

Especially valuable for retrieval networks are some of the features of the packet-switched networks of the ARPANET type. These networks provide efficient multiplexing of communications channels for interactive data that could otherwise use 9 percent or less of channel bandwidth on dedicated channels. Also, intercommunication between interface programs and retrieval systems are provided directly by network procedures and programs. In addition, recent studies [39] have indicated that long-distance communications channel bandwidths and costs will be markedly reduced with satellite technology, making this component of networking even less of a potential barrier to success.

6.2 Effectiveness of Interface Approach

6.2.1 The Dimensions of Effectiveness

We are generally optimistic about the future possibilities of the

virtual-system interface approach and the various techniques we have considered in implementing this approach. One must, however, recognize the several dimensions along which effectiveness can be measured and the tradeoffs that must be weighed between effectiveness and cost.

One dimension is the degree of "virtualness" provided by the interface, that is, the extent to which the interface acts as a common, virtual system, hiding all the heterogeneity and individuality of the different retrieval systems. A second dimension is the completeness with which the interface permits use of the various capabilities of the networked retrieval systems. A third dimension is the exactness of translation; that is, the degree to which the function called for in the common command language is fulfilled, and not overfulfilled, by the translated requests in the different retrieval systems.

Complementing the first three dimensions is the dimension of the comprehensiveness of the totality of retrieval functions permitted through the interface; this, in general, will be greater than what is obtained from the networked retrieval systems since the interface itself provides capabilities not available otherwise. A fifth dimension, closely related to the fourth, recognizes the need for dynamic and integrated character to the solutions: the interface should be extensible as additions to capabilities and other changes ensue and it should be integratable within the larger computer context of distributed networked computation. Finally, the interface may be measured by its simplicity: how easy is it to use by the inexperienced user.

There are, clearly, tradeoffs that may need to be made among the various dimensions and between effectiveness, in general, and cost. For example, exactness of translation and completeness can be increased at the expense of virtualness; in the extreme, a simple transparent mode requires practically no translation and provides access to all the functions of the different retrieval systems at the cost of complexity due to heterogeneity of access for the user.

We believe the approach and techniques we have outline can lead to an interface system that will score high in each of the six dimensional

measures listed above. We discuss below the possible effectiveness of several component techniques we have been considering: a common retrieval language, a Master Index and Thesaurus, and a common bibliographic data structure.

6.2.2 The Common Retrieval Language

The common retrieval language, which has been one of the main foci of interest in this report, has import for all six dimensional measures. We have discussed the ways in which we have tried to make the language simple, extensible, and integratable with other functions. The set of retrieval capabilities outlined in the common command language includes almost all the capabilities of all the retrieval systems we have been working with. In so doing, it includes, a number of capabilities which are not included, at least directly, in that system. Beyond that, there are a number of capabilities that are not included in any of the retrieval systems, either because they are extensions of existing capabilities -- like the extensive storing, sharing, and reusing of searches -- or because they are capabilities peculiar to interface function -- like keeping track of the status of, and connecting to, different systems.

The comprehensive nature of the functions that one would like to obtain through the interface, together with the limited nature of the capabilities available from existing retrieval systems, emphasizes a number of complications that we face in interface building. One problem, of course, is the cost of building into the interface the features themselves or the connections to them. Another problem is that of performing an exact translation of a request in the common command language into one or more commands in a given retrieval system. At a given level of sophistication of the interface the problem may be one of complexity or absolute impossibility.

Thus, for example, as we have seen in Section 2, the current CONIT system cannot translate an arbitrary order of SHOW arguments to the DIALOG language. One could say that the full range of capabilities was available to a user if the user was forced into the complexity

of using the fixed order required by DIALOG. Of course, a slightly more sophisticated interface would handle the question of arbitrary order. Note, however, the default case for set name can be handled only if the interface keeps track of what the number for the current set is, i.e., the translation is dependent on session context. Neither of these examples is particularly difficult to handle, at least conceptually, and the mechanisms for handling them should likely be in any good operational interface. They do point out, however, the idea that there is a series of successively more sophisticated techniques required to handle the problems encountered in achieving higher levels of interface performance along the several effectiveness dimensions.

In a more basic way, however, the translation may be (almost) impossible if the retrieval system cannot perform a given function. For example, a sort of the output by author last name may simply not be possible. The qualifier "almost" is necessary since, e.g., the interface -- at considerable expense, at least compared to the costs of performing this and other operations within the retrieval systems -- could store the output, extract the author names, sort them, and then reorder the output.

Other functions which at least one of the systems we have reviewed cannot do, include: (1) handling of multi-line statements; (2) interrupting; (3) line delete; (4) separate TERSE mode; (5) renaming; (6) automatic stem search; (7) automatic common-word exclusion search (8) Boolean combinations in search statement; (9) nested Boolean statements; (10) unlimited search terms from a user-given stem search; (11) word-order constraints in primary search; (12) record search; (13) display and/or search of thesaurus-related terms; (14) deleting selected sets; (15) reusing a previous search statement; (16) intermixing search statements and combining sets; (17) SDI search; (18) outputting of a selected file; (19) highlighting of matching elements; (20) displaying counts of partial results; (21) saving search sets; (22) "keeping" selected documents in a special set; (23) saving output; and (24) reviewing the previous dialog. It is not too severe, then, to say that

the set of functions that directly and exactly match among the several major retrieval systems is a small subset of the totality of functions. However, we believe -- as we have discussed in part in this report -- that there are methods, more or less difficult, for coming at least moderately close in translating the most basic and important functions from a common language to the different retrieval systems.

6.2.3 The Master Index and Thesaurus

The Master Index and Thesaurus (MAIT), which was mentioned in Section 1.4 and whose specifications were described in detail in our previous report [19], appears to be a powerful tool in providing access to individual, as well as a multiplicity of, data bases. It contains essential information and interrelationships necessary to making intelligent choices of data bases and search strategies.

In particular, a user could make a search request where the search topic is expressed in natural English or in a controlled vocabulary or in some combination of the two. Taking the word stems of the substantive words in the user's request, the interface can use the information in the MAIT -- possibly with the aid of the user -- to find relevant index terms. (We have discussed in our Intrex work [16] the degree of relevance obtainable by these phrase decomposition and stemming techniques.) The document counts associated with these terms provide sound information by which to base a selection of data bases in which to search as well as which index terms under which index elements to search on. Thus, the Master Index and Thesaurus provides the basis for a successful network coupling using natural English words and phrases as a common intermediate language as well as providing greatly enhanced capabilities for access within most existing systems.

These capabilities come at a price: there is a sizable storage requirement and a major updating requirement. However, considering the large potential advantages of the MAIT, neither cost need be thought of as a prohibitive. Index and thesaurus information may be only 5 percent of the total size of a large data base; thus, considering some

overlap in terms, the MAIT for 20 data bases would likely be smaller than a single data base. Also, while updating from multiple sources would require a good deal of coordination, it is possible that most of the advantages of the MAIT could be retained with information that was several months or a year old. This is analogous to a profile for prospective SDI being developed on a retrospective data base.

An indication that Master Index and Thesaurus type concepts are now being recognized and incorporated into current systems is the recent development by Lockheed of its DIALIST [40] merged term frequency indexes in microfiche.

6.2.4 Common Bibliographic Data Structure

Another consideration in the development of means for users to interact effectively with different data bases is the interrelation of the diverse data elements and structures from those data bases. First, searching is done on one or more data elements: in order to translate a search request in the common language into a request in a retrieval system the correct correspondence of data elements must be found. Similarly, user output requests require the specification of combinations of data elements. Finally, in order to combine retrieved document sets from different data bases, we need to: (1) identify when document references from different systems refer to the same document; (2) establish common reference formats; and (3) create common index and catalog data structures.

One part of the solution to these problems is the concept of a common bibliographic data structure mentioned in Section 1.4 with an illustrative example for part of such a structure shown in Fig. 3. We have described the development of this structure in our previous report.[19]

Our recent work has led us to question the relative value of our attempting further efforts in this area at this time. To take the last reason above first, we have not come close to the point of combining document sets from different data bases and creating mini-data-bases with catalog records from them, at least in an online mode. Secondly,

the comparison of data elements for searching may be best handled by the Master Index and Thesaurus, one of whose tasks is to distinguish data elements under which indexed. Thirdly, while a common bibliographic data structure is clearly important if refined distinctions among data elements are to be maintained, we have found that a rough translation among data elements is often all that is possible or needed. For example, subtitles are not usually distinguishable from titles in most data bases and systems in which they are, if anything, simply lumped in with titles. Therefore, we can not easily make use of a structure that is more detailed. At any rate, distinguishing sub-titles from titles may not be very valuable and some systems, as we have seen, do not even allow separation of title from several other data elements.

We note, in any case, that efforts [41-42] appear to be gaining headway to develop a common approach to bibliographic data elements and to data structures in general. It may, then, be advisable in near-term interface work to await these developments while making use of coarse-level common data structures and translations.

6.2.5 Costs and Benefits

It is too early to analyze precisely either the costs or the benefits of the interface approach. However, some order of magnitude estimates can be made. The interface requires duplication of certain functions regularly performed by retrieval systems: the parsing of input requests and the handling of dialog. Also, communications requirements are roughly doubled in that the interface-to-retrieval-system links have to be added to the terminal-to-computer links. Some functions -- like selection of and translation into, target systems -- would be new (although such functions are mirrored in the individual system functions of data base selection and common renaming). On the other hand, the major component function of the actual storage and retrieval from very large data bases would not be required within the interface, at least for a rough translation without a Master Index and Thesaurus. Summing up, we give as a very rough estimate an additional cost for the computer-

system components of approximately 20 percent for the simple interface over those same costs for direct access.

The benefits corresponding to these costs are (1) an increase in accessibility of perhaps an order of magnitude in terms of the number of data bases and systems of practical availability and (2) a reduction of -- and, in some cases, an elimination of -- the need for a trained intermediary information specialist searches. This second benefit has a direct positive benefit in the direction of reducing total costs so that overall costs for interfaced access to retrieval systems could be the sare or less than for direct access. This figure of 20 percent increased computer costs is partially supported by observations on costs of the current CONIT which, although not having all the functions of an operational interface, has been designed more for experimental expediency than efficiency and cost effectiveness.

If we consider a more sophisticated interface with a large Master Index and Thesarus and extensive instructional capabilities, the incremental costs could go to the 50 to 100 percent range, or higher. However, benefits then would include much improved retrieval capability and ability for the end user to make easy access to the data -- allowing many times more users to gain direct access. We would also expect the incremental cost of the interface to be reduced as it became better integrated with the target systems. Of course, this benefit relates to the long-range goal of more compatible retrieval systems.

6.3 Logical Interconnections

As important as how cost effective the interface can be is whether this approach is the appropriate one compared with alternatives, and how it fits into the developing scene of sharing of resources through networking with distributed computation. We have not in the past year seen any reason to believe that the problem of heterogeneous retrieval systems will be resolved in the foreseeable future by any single system becoming dominant nor by existing systems all agreeing to follow a set of (yet-to-be-developed) common standards.

There have been some indications of a trend toward agreements on certain aspects [34,41] of retrieval operations -- like login procedures and data element definitions. Also, of more immediate importance, there is a continuing trend for each system to "fill in the gaps" by incorporating those features which other systems had and it had lacked. Counterbalancing and, perhaps, outweighing these trends are the extensions of these systems in new and different ways and the development of new and different retrieval systems.

Three other paths toward greater compatibility among systems can be stated. We have already mentioned -- above in this section and in Section 6.2.4 -- the efforts toward development of a common bibliographic data structure and common, compatible data structures, in general. These developments can be used by interfaces to aid in providing for greater compatibility; they certainly would not, however, even when they come to fruition, obviate the need to overcome many other differences or to develop networked structures.

A second attempted line of work has been in the area of compatibility among computer programs themselves. It is the goal of this development, either through the use of common or compatible programming languages, to make it easier to transfer programs from one system to another. Interface work should certainly keep track of, and take advantage of these developments. However, major developments along this line do not appear likely to provide important aid for our problems soon; they certainly will not, in themselves, resolve the many problems of networking heterogeneous retrieval systems, especially for existing systems which do not include them.

The third line of progress, which may be the most important in the near and intermediate term, is the development of high-level protocols by which different systems can communicate with each other by user and server processes in a specific application area. As discussed in Sections 4 and 5, these developments are closely related to our common retrieval language development. Several considerations arise in determining the nature of this relationship.

Perhaps the critical questions are how closely the individual systems differ from the protocol and how far the interface should go in creating a fully virtual system that hides, or compensates for, the differences among systems and data bases by major functional capabilities within the interface. To the extent that there are important differences and that much virtualness is desired, we can expect the interface to be a major component of the whole retrieval network. In this case the appropriate structure for networking may be to separate out the large, costly functions into an interface which stands alone -- or has only one or two replications for reliability -- between the various user processes and the server processes encapsulating the retrieval systems.

In such a case, the user and encapsulating processes might be much reduced in scope in that many retrieval functions, as such, would be handled separately by the intermediate interface. In any case, the relations among a common retrieval language, a high-level retrieval protocol, a virtual retrieval system interface, and user and server processes for retrieval and other applications are clearly very important issues in future interface development.

6.4 Areas Requiring Further Work

We have discussed in some detail in this report our approach to the problem of networking heterogeneous retrieval systems and the likelihood of various techniques being useful in the solution of this problem. While we have established a number of avenues that seem fruitful, much additional work is needed to evaluate adequately the cost effectiveness of the individual techniques and the prospects for their successful integration. Because there is such a range and depth of research needed, it is important to select what might be most profitable for near-term effort. We especially want to point out areas within the field of information retrieval that might not otherwise receive adequate attention.

Our immediate plans call for the evaluation in some detail of the question of how effective a fairly simple and not-too-costly interface

(see Section 6.2.5) would be in enhancing access to multiple systems and data bases for the kind of potential user who is both most numerous and most in need of assistance: the ~~inexperienced end~~ user. "Fairly simple and not-too-costly" may be defined roughly as what could ultimately be implemented on a mincomputer class computer. The instructional facilities are clearly key in providing access for inexperienced users. The best way to perform this evaluation in our opinion, is through actual use in an experimental interface.

Subsidiary and longer-range studies, as suggested in the body of this report, are also very important. To reiterate and extend a few of these areas:

(1) Further exploration of the Master Index and Thesaurus concept, including automatic selection of data bases and searches.

(2) Further study of retrieval network software architecture including protocols and user/server programs.

(3) Further analysis of cost/benefits effectiveness, expecially for the more advanced functions of the interface.

(4) Continuing analysis of inter-computer and network communications possibilities.

(5) Extending study of how retrieval systems could be modified, or developed from scratch, so as to perform better in a network environment.

(6) Consideration of the question of whether there are actual advantages to having different retrieval systems.

(7) How is design affected by an operational, many-user environment.

(8) To what extent can the interface development help point toward retrieval standards?

(9) How to integrate the retrieval function via networking into the more general information transfer and general information processing realms.

(10) What special problems will be faced in going from a research to an operational environment; e.g.,

(a) who would administer the interface?

(b) how would payments be handled?

(c) what changes in existing system procedures would be advisable?

6.5 Conclusions

Continued research on the networking of heterogeneous interactive information retrieval systems has lent further credence to the belief in the value of a virtual-system, computer interface as a means to achieve the networking. In part, the evidence for this result has come from the development and initital testing of an experimental interface called CONIT, which contains the basis for a common command and response language and an initital instructional mode. CONIT enables a user to select one of four different retrieval systems to which CONIT automatically connects, and to perform many of the basic retrieval functions of the system using the common language.

Our research has suggested that a practical, operational interface might be developed which would add perhaps 20 percent to the computer costs for online retrieval but relieve the need for a trained intermediary searcher. Such an interface might be most cost effective in the near future if it emphasized access to most, but not necessarily all, existing functions of several retrieval systems for the inexperienced end user.

Progress has also been made in the analysis of the important components of retrieval networks, especially a command language, network structure, and requirements for ease of use. Fruitful areas for additional efforts have been outlined including the study of a number of research issues that have been uncovered but not fully resolved in the work by us and others. These issues include (1) the extension of interface capabilities into a more fully virtual system by such potentially powerful techniques as a Master Index and Thesaurus and (2) the

the design of interface structures so that they fit in with, and enhance, the newly emerging networking software and hardware technologies and other efforts toward compatibility and standardization in retrieval and other information processing areas.

7. PROJECT BIBLIOGRAPHY

1. Therrien, Charles W., Data Communications for an Experimental Information-Retrieval Network Interface, M.I.T. Electronic Systems Laboratory Technical Memorandum ESL-TM-515, August 1, 1973, NTIS Order No. PB 237 975/AS.

2. Marcus, R.S., "A Translating Computer Interface for a Network of Heterogeneous Interactive Information Retrieval Systems", Proceedings of the ACM Interface Meeting on Programming Languages and Information Retrieval (November 1973), SIGIR Forum, Volume IX, No. 3, Winter, 1974, Association of Computing Machinery, pp. 2-12.

3. Reintjes, J.F. and Marcus, R.S., Research in the Coupling of Interactive Information Systems, M.I.T. Electronic Systems Laboratory Report ESL-P-556, June 30, 1974, NTIS Order No. PB 237 974/AS.

4. Marcus, Richard S., "Network Access for the Information Retrieval Application", Panel on Access to Computer Networks, 1975 IEEE Intercon Conference Record, Session 25/4, 1-7 (April, 1975).

5. Marcus, R.S., "Networking Information Retrieval Systems Using Computer Interfaces", Proceedings of the 38th Annual ASIS Conference, October 26-30, 1975. (Volume 12) American Society for Information Science, pp. 77-78.

8. REFERENCES

1. Cuadra, Carlos and Luke, Ann (editors); Annual Review of Information Science and Technology; Volume 10. American Society for Information Science 1975.

2. Cuadra, Carlos; Harris, Jessica; Williams, Martha E; Markuson, Barbara E.; 1965-1975: A Decade of Innovations; General Session I., 38th ASIS Annual Meeting, October, 1975, Boston, Mass.

3. Fano, Robert M., "The MAC System: The Computer-Utility Approach", IEEE Spectrum, January, 1975.

4. Beere, Max P., "Commerical Data Networks Using Available Common Carrier Facilities" in Networks for Research and Education, edited by Martin Greenberger et al, The MIT Press, Cambridge, Massachusetts, pp. 55-63, 1974.

5. Chambers, Jack A. and Poore, Ray V., "Computer Networks in Higher Education: Socio-Economic-Political Factors", Communications of the ACM, 10 (no. 4) 193-199 (April 1975).

6. Roberts, Lawrence G., and Wessler, Barry D., "Computer Network Development to Achieve Pesource Sharing", Proceedings of Spring Joint Computer Conference, AFIPS press, Vol. 36, 543-549 (1970).

7. Wessler, Barry D. and Hovey, Richard B., "Public Packet-Switched Networks", Datamation, July 1974, pp. 85-87.

8. Ornstein, S.M.; Heart, F.E.; Crowther, W.R.; Rising, H.K.; Russell, S.B.; and Michael, A.; "The Terminal IMP for the ARPA Computer Network", AFIPS Proceedings, Spring Joint Computer Conference, Vol. 40, 1972. pp. 243-254.

9. Bouknight, W.J.; Grossman, G.R.; and Grothe, D.M.; "The ARPA Network Terminal System -- a New Approach to Network Access", Procéedings DATACOM 1973. pp. 73-79.

10. Retz, David L., "ELF -- a System for Network Access", 1975 IEEE Intercon Conference Record, Session 25/2, 1-5 (April, 1975).

11. Rosenthal, Robert, "Accessing Online Network Resources with a Network Access Machine", IEEE Intercon Conference Record, Session 25/3, (April, 1975).

12. Crocker, Stephen D.; Heafner, John F.; Metcalfe, Robert M.; and Postel, Jonathan B.; "Function-Oriented Protocols for the ARPA Computer Network", AFIPS Proceedings, Spring Joint Computer Conference, Vol. 40, 1972. pp. 271-279.

13. Thomas, Robert H., "A Resource Sharing Executive for the ARPANET", AFIPS Conference Proeedings, Vol. 42, June 1973, pp. 359-367.

14. Balzer, Robert; Cheatham, T.E.; Crocker, Stephen; and Warshall Stephen; The National Software Works, University of Southern California, Information Sciences Institute Memorandum, December 20, 1973.

15. McCarn, Davis B., "Trends in Information", Information Utilities: Proceedings of the 37th ASIS Annual Meeting, October 13-17, 1974. American Society for Information Science, pp. 145-150.

16. Overhage, C.F.J. and Reintjes, J.F., "Project Intrex: A General Review", Information Storage and Retrieval, 10 (no. 5) pp. 157-188 (1974).

17. Benenfeld, Alan R.; Pensyl, Mary E.; Marcus, Richard S.; and Reintjes, J.F.; NASIC at M.I.T. Final Report, M.I.T. Electronic Systems Laboratory Report ESL-FR-587, February 28, 1975.

18. Wanger, Judith; Fishburn, Mary and Cuadra, Carlos A.; On-Line Impact Study, Summary Report, System Development Corporation Report, December, 1975.

19. Reintjes, J.F. and Marcus, R.S., Research in the Coupling of Interactive Information Systems, M.I.T. Electronic Systems Laboratory Report ESL-R-556, June 30, 1974, NTIS Order No. PB 237 974/AS.

20. Therrien, Charles W., Data Communications for an Experimental Information-Retrieval Network Interface, M.I.T. Electronic Systems Laboratory Technical Memorandum ESL-TM-515, August 1, 1973, NTIS Order No. PB 237 975/AS.

21. Walker, Donald E., (editor), Interactive Bibliographic Search: The User/Computer Interface, AFIPS Press, 1971.

22. Heafner, J.F.; Protocol Analysis of Man-Computer Languages: Design and Preliminary Findings, University of Southern California, Information Sciences Institute Report ISI/RR-75-34; July, 1975. (NTIS Order No. AD-A013 568).

23. Moghdam, Dineh, "User Training for On-Line Information Retrieval Systems", Journal of the American Soceity for Information Science, Vol. 26, no. 3 (May 1975) pp. 184-188.

24. Boies, Stephen J., User Behavior on an Interactive Computer System, International Business Machines, Thomas J. Watson Research Center, Interim Technical Report No. RC 4169; January, 1973. (NTIS Order No. AD-754 836).

25. MULTICS Programmers' Manual, Reference Guide and Commands and Active Functions; Series 60 (Level 68), Honeywell Information Systems Inc., December, 1975.

26. Kennedy, T.C.S., "Some Behavioral Factors Affecting the Training of Naive Users of Interactive Computer Systems", International Journal of Man-Machine Studies, Vol. 7, No. 6 (November, 1975) pp. 817-834.

27. Wilks, Yorick, "National Language Understanding Systems within the A.I. Paradigm: A Survey and Some Comparisons", American Journal of Computational Linguistics, 1976, No. 1. Microfiche 40, Card 103.

28. Schank, Roger C. and Nash-Webber, Bonnie L. (editors), Theoretical Issues in Natural Language Processing, An Interdisciplinary Workshop in Computational Linguistics, Psychology, Linguistics, and Artificial Intelligence, 10-13 June 1975; Cambridge , MA.; Center for Applied Linguistics; Arlington, Virginia

29. Diller, Timothy C. (Editor), Proceedings of the 13th Annual Meeting of ACL, in American Journal of Computational Linguistics 1975 No. 4, Microfiches 32-36, Cards 85-89.

30. Shapiro, Stuart C. and Kwasny, Stanley C., "Interactive Consulting via Natural Language", Communications of the Association for Computer Machinery, Vol. 18, No. 8, August, 1975. pp. 459-462.

31. Kugel, Peter, "Dirty Boole?" Journal of the American Society for Information Science, Vol. 22, No. 4 (July, 1971) pp. 293-294.

32. Marcus, R.S., Benenfeld, A.R., and Kugel, P., "The User Interface for the Intrex Retrieval System", in Interactive Bibliographic Search: The User/Computer Interface, edited by D.E. Walker, AFIPS Press, 1971, pp. 159-201.

33. Martin, Thomas H.; A Feature Analysis of Interactive Retrieval Systems, Stanford University, Institute for Communication Research, Report SU-COMM-ICR-74-1, September, 1974.

34. Neumann, Albrecht J.; A Basis for Standarization of User-Terminal Protocols for Computer Network Access, National Bureau of Standards, Technical Note 877; July, 1975.

35. Palme, Jacob; Interactive Software for Humans, Research Institute of National Defense (Sweden), Report F)A C10029 M3(E5); July, 1975.

36. Collins, Alan M.; Passafiume, Joseph J.; Gould, Laura, and Carbonell, Jaime G.; Improving Interactive Capabilities in Computer-Assisted Instruction, Bolt, Beranek and Newman, Inc. Report BBN No. 2631, 1973.

37. Penniman, W. David, "A Stochastic Process Analysis of On-Line User Behavior", Proceedings of the 38th Annual ASIS Conference, October, 1975 (Volume 12) pp. 147-148.

38. Bolt, Beranek and Newman; "MSG: The Interprocess Communications Facility for the National Software Works", BBN Report No. 3237; January 23, 1976. (Also, Massachusetts Computer Associates Document No. CADD-7601-2611).

39. Corte, Arthur B. and Pool, Ithiel de Sola; "International Data Communication Capabilities and the Information Resolution", Proceedings of the 38th Annual ASIS Conference, October, 1975 (Volume 12) pp. 1-2.

40. Lockheed Information Retrieval Service, Brochure on DIALIST, Lockheed Palo Alto Research Laboratory, Lockheed Information Systems; October, 1975.

41 Williams, Martha E., Preece, Scott, E.; and Rouse, Sandra H.; Data Element Analysis and Use of a Relational Data Base Structure for Mapping Bibliographic and Numeric Data Bases, Information Retrieval Research Laboratory, University of Illinois. Also presented at the National Bureau of Standards Second National Symposium on the Management of Data Elements in Information Processing, Gaithersburg, Maryland, 24 October 1975.

42. ACM-SIGMOD Workshop on Data Description, Access and Control, Ann Arbor, May 1-3, 1974; Association for Computer Machinery (Special Interest Group on the Management of Data). New York (1974).

APPENDIX A

SAMPLED USER/CONIT DIALOG

This appendix lists excerpts from dialog between a user and CONIT which are intended to illustrate various facets of the interface situation. The excerpts are reproduced from computer terminal printouts with some reduction in size. Annotations by the authors have been added to help reader understanding and are enclosed in boxes. Excerpts form a continous dialog except where ellipses (...) indicate some dialog has been taken out. Each different session is so indicated.

The first three pages of excerpts (A2-A4) show a session with a very simple round of selecting systems, data bases, performing searches, and getting output. Latter sessions (pages A5-A20) explore some of the more involved considerations.

The first page is additionally annotated to show the origin and processing of the messages. User commands are underlined. Messages originating from CONIT have a single line alongside them in the margin. Messages originating from a retrieval system that have been translated (at least in part) are indicated by a double line in the margin. Messages without any markings originated in retrieval systems and were passed through by CONIT with no translation.

conit
Welcome to CONIT. For help on how to use CONIT you may type 'help' followed by a carriage return; otherwise, you may now type any CONIT command.
USER::
pick suny
Attachment successful.

INDICATES SUCCESS IN ATTACHING TIP PORT

SUNY/MEDLINE:
Connection completed.

INDICATES PROPER INITIAL RESPONSE SEEN

SS 1 is number for your next search set.

suny is connected successfully
You are now speaking in CONIT

INDICATES LOGIN COMPLETED

(THESE MESSAGES, AND OTHERS, COULD BE IN BETTER, MORE VIRTUAL FORMAT AS DISCUSSED IN REPORT)

USER::
find radiation

SUNY/MEDLINE:
Your search resulted in set1 which contains this many documents: (2684)

SS 2 is number for your next search set.

USER::
show title docs1-3

SUNY/MEDLINE:

1
TI - :STATISTICAL EVALUATION OF LIGHT PROTECTION FACTORS:

2
TI - PROGNOSIS AND POST-THERAPEUTIC FOLLOW-UP OF BREAST CANCERS BY THERMOGRAPHY.

3
TI - MICROWAVE THERMOGRAPHY: A METHOD OF DETECTING SUBSURFACE THERMAL PATTERNS.

SS 2 is number for your next search set.

USER::
pick lms

LOCKHEED DIALOG SYSTEM PICKED

sent

INDICATES LOGIN SENT

HOST IS ONLINE
$$$$$$$$ RECONNECT @ 15:53:05
)
NEW FILE - DISSERTATION ABS. - SEE ?NEWS

USER::
pick data eric

FILE1

```
File reset: ERIC FULL-TEXT ED & EJ DEC-7

USER::
find radiation

Your search resulted in set2  454 RADIATION  (PROCESS OF ENERGY

USER::
show docs@show title docs1-3

Msg from 1:DIALOG GOING DOWN FOR THE DAY
  IN 5 MINUTES
1
EJ121600
  Solar Heated Homes: They're Here

2
EJ121508
  A Course in Nuclear Radiation for All High School Students

3
EJ121506
  Energy Alternatives

USER::
pick sdc
sent
;
YOU ARE ON LINE L8A

HELLO FROM SDC/ORBIT.
YOU ARE NOW CONNECTED TO THE ORBIT DATABASE.

SDC/ORBIT:
****
TODAY ONLY: GEOREF AND APIPAT WILL NOT BE AVAILABLE. SORRY
FOR THE INCONVENIENCE.
****

SS 1 is the number of your next  SDC/ORBIT search set.

USER::
pick data ntis

SDC/ORBIT:
-.
THE TIME IS NOW  6 :56  P.M. (EST), 01/16/76
YOU ARE NOW CONNECTED TO THE NTIS DATABASE.

SS 1 is the number of your next SDC/ORBIT search set.

USER::
find radiation
```

@ CANCELS LINE TO THAT POINT

NOTE SPECIAL AND IMPORTANT MESSAGE IN LOGIN

SDC ORBIT SYSTEM PICKED

NOTE SPECIAL MESSAGE

```
SDC/ORBIT:
SS 1 is resulting set containing this many documents:(21223)

SS 2 is the number of your next SDC/ORBIT search set.

USER::
show title docs1-3


SDC/ORBIT:

TI  - Results of the Lyman alpha Measurements of the  Satellite Dial
      Ergebnisse der Lyman-alpha  Messungen des Satelliten Dial

TI  - Materials  Data Retrieval at Estec

TI   - Angular Distributions of Electrons of Energy Esub E Greater Than
       0.06 Mev in the J ovian Magnetosphere

SS 2 is the number of your next SDC/ORBIT search set.

USER::
pick nlm
sent
;
 LOGON IN PROGRESS AT 18:59:30 ON JANUARY 16, 1976
 *** BACK72  AT NLM AND SUNY CONTAINS 1972 CITATIONS ONLY.***
 TOTAL ACTIVE TSO USERS: 06
READY


TSO LINE 07E
ELHILL3 IS NOT AVAILABLE AT THIS TIME.
READY

USER::
pick suny
Can't log off nlm

USER::
sp host
Transparent mode.  All of your input will now be sent directly to nlm without
interpretation,except the 'speak conit' command which will cause CONIT  to
resume interpreting your requests.

USER::
logoff


      LOGGED OFF TSO AT  19:03:37 ON JANUARY 16, 1976+

DROPPED BY HOST SYSTEM
PLEASE LOG IN:
USER::
suny4


PASSWORD:bcn
```

NOTE IMPORTANT MESSAGE

CONIT DOES NOT YET RECOGNIZE UNAVAILABILITY MESSAGE AND GETS CONFUSED. ANALYST GOES INTO TRANSPARENT MODE TO CARRY OUT TASK "MANUALLY" (WITHOUT TRANSLATION)

• • •

NEW SESSION

```
PICK UNTAC
REQUEST FOR CONNECTION REFUSED BY FOREIGN HOST.
PORT 13 UNAVAILABLE, TRYING NET PORT
ATTACHMENT SUCCESSFUL.
YOU ARE NOW SPEAKING TO CONIT
```

SECOND PORT TRIED WHEN ONE UNAVAILABLE

```
USER:>
LS ALL
```

LIST STATUS

```
THE SYSTEM CURRENTLY SELECTED IS:  UNTAC
OTHER SYSTEMS CURRENTLY LOGGED IN:
NONE
YOU ARE NOW SPEAKING TO CONIT LANGUAGE
YOUR DIALOG WITH CONIT IS NOW IN SCREEN - CREOLE - CLEAR MODE
THE FOLLOWING 'IT' PORTS ARE ATTACHED:
CCR PORT 13
THE CAPTAIN ... IS: NONE

USER:>
LT
TO LIST:>
SHOW LISTS PLAIN IN SUBJECT>
SHOW LITRN
PRINT OUTPUT>
PICK SETS>
PICK DATABASE PLAIN IN PATH>
OR SETOR 3>
FIND AUTHOR=AUTHOR>
FIND SUBJECT>
COMBINE SETS>
AND SET AND 2>
AND NOT SET AND NOT 3>
 DOCUBE PLAIN IN DOCS>

USER:>
FIND RADIATION EFFECTS
```

LIST COMMAND TRANSLATION TABLE NOTES: (1) OLDER VERSION THAN IN APPENDIX C; (2) NO RESPONSE TRANSLATION TABLE AT THIS POINT

```
I SEARCH ON YOUR REQUEST "SUBJECT RADIATION EFFECTS" FOUND 2
DOCUMENTS THAT ARE NOW YOUR CURRENT LIST.
YOUR NEW CURRENT LIST IS NAMED 31

READY

USER:>
SHOW

1. DOCUMENT 6A0: CURRENT TRENDS IN THE CONTROL AND SURVEILLANCE OF
PRESSURE VESSEL IRRADIATION EFFECTS; CHERNOCK, W.P.; KOZIOL, J.J.

2. DOCUMENT  01: LIGHT-INTENSITY CONTROL DEVICE UTILIZING ORIENTED
PARTICLES SUSPENDED IN A CELL; SAUVER, G.F., (INVENTOR)

READY

USER:>
```

```
SHOW MATCH

1. DOCUMENT 1 -

 MATCH:
  SPEC-EXPRESSION:
   PRESSURE VESSEL IRRADIATION EFFECTS
  HEADING-OR-SUBHEADING:
    PHYSICAL EFFECTS OF RADIATION
    RADIATION SHIELDING
    RADIATION EFFECTS
  MAIN-TITLE:
    CURRENT TRENDS IN THE CONTROL AND SURVEILLANCE OF PRESSURE
      VESSEL IRRADIATION EFFECTS

2. DOCUMENT 501

  AUTH:
  CORPORATE-MAIN-HEADING:
    GENERAL ATOMIC CORP.
  ABSTRACT-SUMMARY:
    THE LIGHT TRANSMITTED BY OPTICAL DEVICE IS CHANGED BY MEANS OF A
      REFLECTION BLIND EFFECT WITH THE ELEMENTS BEING PROVIDED
      WITH A PARTITION WHICH IS LOCATED IN A DIRECTION TRANSVERSE TO
      THE PARTICLE ORIENTATION SO THAT THE DEL MAY BE DEFORMED TO
      PRESENT AN INCREASING PORTION OF A PARTICLE CROSS SECTION
      TO LIGHT RAYS WHICH OTHERWISE WOULD PASS THROUGH THE DEVICE
      UNOBSTRUCTED.
  SPEC-EXPRESSION:
    REFLECTION BLIND EFFECT

READY

USER:

FIND AUTHOR LIBRARY

    1 DOCUMENT SATISFIED AUTHOR CRITERIA. FOUND 1 DOCUMENT THAT
IS NOW YOUR CURRENT LIST.
YOUR NEW CURRENT LIST IS NAMED 2

READY

USER:
COMBINE SET 1 AND SET 2

THERE ARE 1 DOCUMENTS ON YOUR NEW CURRENT LIST.
YOUR NEW CURRENT LIST IS NAMED 3

READY

USER:
SHOW TITLE


1. DOCUMENT 1 - 3

 TITLE:
   CURRENT TRENDS IN THE CONTROL AND SURVEILLANCE OF PRESSURE VESSEL
   IRRADIATION EFFECTS

READY
```

INDEXED EXPRESSIONS THAT CONTAIN RADI+ OR EFFECT+ STEM

USER:
SHOW ALL

1. Document 44

MAIN-TITLE:
CURRENT TRENDS IN THE CONTROL AND SURVEILLANCE OF PRESSURE VESSEL IRRADIATION EFFECTS

SOURCE:
Physical Effects
Electrical and Electronic Effects
Computer and Control Systems
FEC

ACCESSION-RECORD-NUMBER:
T223126
T219315
T211113

CLASSIFICATION-NUMBER:
12.47
42.30
73.50

CLASSIFICATION-SYSTEM:
1 NRC Unified Classification
1 NRC Unified Classification
1 NRC Unified Classification

HEADING-OR-SUBHEADING:
NUCLEAR REACTORS, FISSION
PHYSICAL EFFECTS OF RADIATION
RADIATION MONITORING
NUCLEAR REACTORS
RADIATION EFFECTS
NUCLEAR REACTORS

HEADING-OR-SUBHEADING-TYPE:
Main Heading
Main Heading
Main Heading
Main Heading
Main Heading
Main Heading

FREE-EXPRESSION:
VESSEL MATERIAL FRACTURE TOUGHNESS
NEUTRON EXPOSURE
CHARPY V SAMPLES
CURRENT TRENDS
CONTROL
SURVEILLANCE
PRESSURE VESSEL IRRADIATION EFFECTS
TRANSITION TEMPERATURE

CONTENT:
DEVELOPMENT OR APPLICATION

ABSTRACT-SENTENCE:
THE PAPER DESCRIBES A TYPICAL CURRENT PRESSURE VESSEL IRRADIATION SURVEILLANCE PROGRAMME.
THE CASES OF A REACTING SIGNIFICANT REDUCTIONS IN IRRADIATION INDUCED EMBRITTLEMENT TEMPERATURE ... ARE ACHIEVABLE THROUGH CONTROL OF RESIDUAL ELEMENT CONTENT IN BOTH PLATES AND WELDS ARE DISCUSSED.
PROBABLE TRENDS IN LICENSING REQUIREMENTS ARE OUTLINED.

ABSTRACT-PARAGRAPH-NUMBER:
1
1
1

LAST-NAME:
CHOPCHOCK
POZIOL

INITIALS:
M.F.
J.J.

PERSONAL-NAME-RELATOR:
AUTHOR
AUTHOR

AFFILIATION-MAIN-DIVISION:
COMBUSTION ENGNG. INC., WINDSOR, CONN., USA
COMBUSTION ENGNG. INC., WINDSOR, CONN., USA

AFFILIATION-TYPE:
PERSON AFFILIATION
PERSON AFFILIATION

CORPORATE-MAIN-HEADING:
ILL. INST. TECHNOL., AMERICAN SOC. MECH. ENGRS., AMERICAN NUCLEAR SOC., ET AL

CORPORATE-RELATOR:
SP

MEETING-NAME:
PROCEEDINGS OF THE AMERICAN POWER CONFERENCE

MEETING-PLACE:
CHICAGO, ILL., USA

MEETING-DATE:
20-22 APR 1971

MEETING-RELATOR:
PC

DOCUMENT-RELATION-TYPE-CODE:
01

CITATION-ELEMENT-TYPE-CODE:
60

```
CITATION-AUTHOR-TITLE-OR-NUMBER-ELEMENT:
 PROCEEDINGS OF THE AMERICAN POWER CONFERENCE

PUBLICATION-DATE-OF-RELATED-DOCUMENT:
 1971

RELATED-DOCUMENT-IMPRINT:
 ILL. INST. TECHNOL., CHICAGO, ILL., USA

RELATED-DOCUMENT-COLLATION:
 275-83

OTHER-CITATION-ELEMENTS-CODE:
 60

OTHER-CITATION-ELEMENTS:
 XLU + 1177

RELATED-DOCUMENT-FORM-CODE:
 10

INSPEC-CONTROL-NUMBER:
 384944

INSPEC-RECORD-TYPE:
 06

PLACE-OF-PUBLICATION:
 CHICAGO, ILL., USA

NUMBER-OF-PAGES:
 XLU + 1177

PUBLISHER:
 ILL. INST. TECHNOL.

READY

USER::
```

• • •

NEW SESSION

```
pick medline
Attachment successful.
Logon to host started.

 LOGON IN PROGRESS AT 14:56:52 ON JANUARY 14, 1976
NO BROADCAST MESSAGES
Response not yet recieved from medline
Shall I continue listening?
Type yes or y to continue, no or n to stop or dis to disconnect host.
y
 TOTAL ACTIVE TSO USERS: 44
READY

TSO LINE OFA

HELLO FROM ELHILL 3.
YOU ARE NOW CONNECTED TO THE MEDLINE FILE.


MEDLINE:

SS 1 is the number for your next MEDLINE search set.
You are now speaking in CONIT

USER::

show data


MEDLINE:
YOU MAY ACCESS THE MEDLINE, SDILINE, CATLINE, MESH VOCABULARY,
JOURNAL AUTHORITY, NAME AUTHORITY, OLD MESH VOCABULARY, AVLINE,
CANCERLINE, CHEMLINE, TOXLINE, CANCERPROJ AND EPILEPSY FILE SETS.
YOU ARE NOW CONNECTED TO THE MEDLINE FILE.

SS 1 is the number for your next MEDLINE search set.

USER::
pick data sdiline


MEDLINE:

 35 USERS LOGGED IN PRESENTLY.
YOU ARE NOW CONNECTED TO THE SDILINE FILE.

SS 1 is the number for your next MEDLINE search set.

USER::
show index radiation


MEDLINE:

POSTINGS    TERM
     1      RADIATE (TW)
     4      RADIATING (TW)
     2      RADIATION (MH)
   156      RADIATION (TW)
    23      RADIATION CHIMERA (MH)
To see more type 'show more'.
```

```
USER::
show more


MEDLINE:

POSTINGS     TERM
    37       RADIATION DOSAGE (MH)
   167       RADIATION EFFECTS (MH)
    26       RADIATION GENETICS (MH)
    20       RADIATION INJURIES (MH)
    14       RADIATION INJURIES, EXPERIMENTAL (MH)
To see more type 'show more'.

USER::
find radiation+                          USER-GIVEN-STEM SEARCH

MEDLINE:
Your search resulted in set1 which contains this many documents: (349)

SS 2 is the number for your next MEDLINE search set.

USER::

                                  • • •

find skin+


MEDLINE:
Your search resulted in set2 which contains this many documents: (700)

SS 3 is the number for your next MEDLINE search set.

USER::

                                  • • •

find tissue+


MEDLINE:
Your search resulted in set3 which contains this many documents: (1057)

SS 4 is the number for your next MEDLINE search set.

USER::
combine set3 a#or set2          # CANCELS PREVIOUS CHARACTER


MEDLINE:
Your search resulted in set4 which contains this many documents: (1665)

SS 5 is the number for your next MEDLINE search set.

USER::
combine set1 and set4


MEDLINE:
Your search resulted in set5 which contains this many documents: (61)

SS 6 is the number for your next MEDLINE search set.

USER::
```

show title docs1-3

MEDLINE:

1
TI - ¢<Statistical evaluation of light protection factors¢>

2
TI - Prognosis and post-therapeutic follow-up of breast cancers by thermography.

3
TI - Microwave thermography: a method of detecting subsurface thermal patterns.

SS 6 is the number for your next MEDLINE search set.

USER::
show abstract docs1-1

MEDLINE:

1
AB - The protection against erythema belongs to the cosmetic effects which lend themselves to mathematical treatment. It is demonstrated -- on the basis of the optimal definitions given by Ellinger and Schulze -- that the calculation of the mean value of the light-protection factor Q as hitherto in use, does not correspond to the real frequency-distribution. On the contrary there exists, independent of the radiation source having sunlike characteristics and of the distance from the radiator, a binary-logarithmic standard distribution. With reference to the gradation principles of the human skin a transformation of the pertinent differences of area is necessary first, i.e. a transformation responding to the Caussian standard distribution principle. Tables are presented concerning the transformation and the practical evaluation of the light-protection factor Q. By aid of these tables a standardization of the factors Q measured by different authors has been attained as well as a standardized statistical-mathematical analysis. The investigation of the threshold dose producing erythema on the unprotected human skin has revealed a superposition of three frequency-distribution types (showing logarithmic distribution, too) having different standard deviations. The results of this entirely statistical classification permit a safe forecast: the sunburn protection inherent in the human skin is compounded of several contributing factors which are interconnected multiplicatively, not additively.

SS 6 is the number for your next MEDLINE search set.

USER::
st system

• • •

SPECIAL TRANSLATION TABLE SET FOR TEST

• • •

USER::

open skinrad
File has been opened.

OPEN FILE NAMED SKINRAD FOR SAVING

USER::
save

SAVE (=FILE) NEXT OUTPUT IN SAVED FILE

USER::

st med

RESET TO REGULAR MEDLINE TRANSLATION TABLE

USER::
show title docs1-2

MEDLINE:

1
TI - ¢<Statistical evaluation of light protection factors¢>

2
TI - Prognosis and post-therapeutic follow-up of breast cancers by thermography.

SS 6 is the number for your next MEDLINE search set.

USER::

• • •

pick data cancerline

MEDLINE:

27 USERS LOGGED IN PRESENTLY.
YOU ARE NOW CONNECTED TO THE CANCERLINE FILE.

SS 1 is the number for your next MEDLINE search set.

USER::

• • •

find skin and radiation

MEDLINE:
Your search resulted in set2 which contains this many documents: (289)

SS 3 is the number for your next MEDLINE search set.

USER::

• • •

show 1

MIXED CONIT AND ORBIT COMMAND

MEDLINE:

1
AU - Sheleshko PV
TI - CLINICAL, HISTOLOGICAL AND HISTOCHEMICAL DIFFERENTIATION OF PRECANCEROUS CONDITIONS OF THE SKIN.
SI - CARC/74/03302
SO - Vestn Dermatol Venerol; 10):24-28 1974

SS 3 is the number for your next MEDLINE search set.

USER::
st system

USER::
show@save

USER::
st med

USER::
show title docs1-3

MEDLINE:

1
TI - CLINICAL, HISTOLOGICAL AND HISTOCHEMICAL DIFFERENTIATION OF PRECANCEROUS CONDITIONS OF THE SKIN.

2
TI - MINIREVIEW. REPAIR OF DNA IN MAMMALIAN CELLS.

3
TI - IMMUNOLOGIC ABNORMALITIES IN HEAD AND NECK CANCER.

SS 3 is the number for your next MEDLINE search set.

USER::
st m#system.

USER::
view skinrad
skinrad contains 29 lines .

USER::
view lines 1-29

REVIEW SEARCHES IN 2 DATA BASES IN SAVED FILE

MEDLINE:

1
TI - ¢<Statistical evaluation of light protection factors¢>

2
TI - Prognosis and post-therapeutic follow-up of breast cancers by thermography.

SS 6 is the number for your next MEDLINE search set.
USER:

MEDLINE:

1
TI - CLINICAL, HISTOLOGICAL AND HISTOCHEMICAL DIFFERENTIATION OF PRECANCEROUS CONDITIONS OF THE SKIN.

2
TI - MINIREVIEW. REPAIR OF DNA IN MAMMALIAN CELLS.

3
TI - IMMUNOLOGIC ABNORMALITIES IN HEAD AND NECK CANCER.

SS 3 is the number for your next MEDLINE search set.
USER:

USER::
show news
show is not a legal CONIT command.
Type 'explain commands' for a list of commands.

WRONG TRANSLATION TABLE SET.

USER::
st med

```
USER::
show news

 'MEDLINE.NEWS.DATA'
14 JAN  - IN AVLINE PRINT DETAILED DOES NOT AND WILL NOT PRINT
      ABSTRACTS.  THE EXPLAIN UNIT RECORD IS IN ERROR AND WILL
      BE CORRECTED.
*****
13 JAN - UNTIL FURTHER NOTICE, IN THE AVLINE FILE, TEXT
      WORD SEARCHING IS NOT AVAILABLE ON CORPORATE NAMES
      AND SERIES TITLES.
*****
12 JAN - AVLINE WILL BE AVAILABLE JAN  13 AT NLM ONLY; AND NOT
      JAN 12 AS PREVIOUSLY ANNOUNCED.
*****
12 JAN - SDILINE AT NLM AND AT SUNY NOW CONTAINS FEB 76
      CITATIONS.  MEDLINE AT NLM AND MEDLINE AT SUNY NOW
      CONTAINS 76 CITATIONS FROM JAN 1974 THRU FEB 1976.
      BACK72, AVAILABLE THRU OFFSEARCH AT NLM AND SUNY,
      NOW HAS 1972 AND 1973 CITATIONS IN THE DATA BASE.
      THE 1976 MESH SHOULD NOW BE USED WHEN SEARCHING AT
      NLM OR SUNY.
*****
6 JAN  - THE EPILEPSY DATA BASE IS NOW AVAILABLE TO ALL
     U.S.  MEDLINE AND TOXLINE USERS. ENTER "FILE EPILEPSY.
     FOR  SEARCHABLE ELEMENTS ENTER "EXPLAIN UNIT RECORD.
     THE SEARCHING DEFAULT IS TO ALL. FILE CONTAINS 16231
     RECORDS FROM 1945 TO 1973.
*****
IF YOU HAVE TROUBLE USING ON-LINE FILES AT NLM OR SUNY, NOTIFY
  MS GRACE H MCCARN, MEDLARS MANAGEMENT SECTION (301/496-6193).
  EVENINGS CALL THE NLM COMPUTER ROOM (301/654-6422), OR
  THE SUNY COMPUTER ROOM (518/474-2921).
TSO LINE OFA

MEDLINE:

SS 3 is the number for your n ext MEDLINE search set.

USER::
show data all

 'MEDLINE.FILES.DATA'
  DATA BASE    TOTAL RECORDS     ENTRY DATES      COVERAGE/CURRENCY

 *BACK68           545,463      651113-681111     JAN 68 - DEC 68
  BACK69           649,346      681117-711117     JAN 69 - DEC 71
  BACK72           449,361      711130-731116     JAN 72 - DEC 73
  *CATLINE         155,277                        1965 -  9  JAN 1976
 *CANCERLINE        45,383                        JAN  63 - DEC 74
 *CANCERPROJ         5,517                        1974 - 1975
 *CHEMLINE          76,955
 *EPILEPSY          16,231                        1945 - 1973
 *JOURNAL AUTH       4,213                        1974
  MEDLINE (NLM)    486,937      731130-760102     JAN  74 - FEB 76
  MEDLINE (SUNY)   486,937      731130-760102     JAN 74 - FEB  76
  MESH.VOC          13,624                        1975
  *NAME AUTH
  SDILINE (NLM)     21,138      751210-760102     FEB 76
  SDILINE (SUNY)    21,138      751210-760102     FEB  76
  *TOXLINE.        294,013
    CBAC           146,805                        1971 - MID-DEC 75
    TOXBIB          64,007                        1971 - DEC 75
    IPA             21,088                        1971 - SEPT 75
    HEEP            44,504                        1971 - SEPT 75
    NAPAB/PESTAB    10,251                        1971 - JUNE 75
```

```
    EMIC          7,358                    1971 - 1974
  *TOXBACK      186,248
    CBAC         90,922                    1965 - 1970
    TOXBIB       60,229                    1966 - 1970
    IPA           8,594                    1970
    HEEP          3,474                    1972
    HAPAB         7,221                    1966 - 1970
    EMIC          5,765                    1968 - 1970
    HAYES        10,043                    1930 - 1970
*****

1) * = FILES AVAILABLE AT NLM ONLY.
2) THE BACKFILES ARE AVAILABLE ONLY THRU OFFSEARCH.
3) TOXBACK IS AVAILABLE ONLY THRU OFFSEARCH AT NLM.
TSO LINE OFA

MEDLINE:

SS 3 is the number for your next MEDLINE search set.

USER::


speak moniotr

USER::
speak monitor
From CONIT:
You are now speaking in MONITOR mode          MONITOR MODE

USER::
pick data medline
From CONIT:
sent
From CONIT  to medline
"USERS""FILE MEDLINE



From medline:
¢000¢023¢021          OCTAL CODES OF FORMAT CHARACTERS

From medline:

¢023¢021¢201

From medline:

PROG:                 RESPONSE FROM MEDLINE
MEDLINE:
From medline:         TRANSLATED RESPONSE FOR USER

MISSING DOUBLE-QUOTE MARK.

From medline:

 37 USERS LOGGED IN PRESENTLY.
 37  USERS LOGGED IN PRESENTLY.
From medline:

YOU ARE NOW CONNECTED TO THE MEDLINE FILE.
YOU ARE NOW CONNECTED TO THE MEDLINE FILE.
From medline:
```

• • •

pick lms
sent

NEW SESSION
LOCKHEED DIALOG PICKED

HOST IS ONLINE
$$$$$$$$ LOGON 3 7:36:13

FILE 32 (METADEX) ONLINE
FILE 35 (DISSERTATIONS) ONLINE

USER::
show data

```
 1 -ERIC: ED, EJ
 3 -CHEMICAL ABSTRACTS CONDENSATES
 4 -EXCEPTIONAL CHILDREN ABST.
 5 -BIOSIS PREVIEWS
 6 -NTIS
 7 -SOCIAL SCISEARCH
 8 -COMPENDEX (EI)    9 - AIM/ARM
10 -NAL/CAIN          11 -PSYCH ABS
12 -INSPEC-PHYSICS
13 - INSPEC-ELECTRONICS/COMPUTERS
14 -ISPEC             15 -ABI/INFORM
16 -PTS CHEM/ELECT. MKT. ABST.
17 - PTS WEEKLY CMA, ERA, AND F&S
18 -PTS F&S           19 - PTS CIN
20 -PTS DOM. STAT     21 -PTS FOR. STAT
22 -EISYour search resulted in set 23 -CLAIMS-CHEMICAL
24 -CLAIMS-GEN
26 -FOUNDATION DIRECTORY
27 -FOUNDATION GRANTS INDEX
28 -OCEANIC ABS       29 -METEOR/GEO ABS
32 -METADEX
34 -SCISEARCH         35 -DISSERTATIONS
```

USER::
pick data ntis

USER::
*

.FILEG
Event: Time,SearchTime,Date,User#,Descr,Docs,File
End: 7:49:25,013.21,01/19/76,0108,0000,0000,01
File reset: NTIS 1964-1976 ISS 02

USER::
show index skin

Ref	Index-term	Type	Items	RT
E1	SKILLS CONVERSION PROJECT		21	
E2	SKIN		1	
E3	SKINNER		19	
E4	SKINNERS		40	
E5	SKINNING		11	
E6	SKIN		2231	
E7	SKIN (ANATOMY)		28	
E8	SKIN (ANATOMY) 24		1	

```
E9    SKIN (BIOL)----------          8
E10   SKIN (STRUCTURAL MEMBER)      29
E11   SKIN ABSORPTION---------       8
E12   SKIN ANALYSORS----------       1
E13   SKIN BEADS-------------        1
E14   SKIN CANCER-------------        3
E15   SKIN DISEASES-----------       57
E16   SKIN EFFECT-------------       16
E17    SKIN FRICTION-----------     567
E18   SKIN FRICTION DRAG------       3
E19   SKIN FRICTION GAGES-----.      1
For more type 'show more'

USER::
find sin##kin


Your search resulted in set1    2231 SKIN

USER::
find radiation


Your search resulted in set2 35305 RADIATION

USER::
combine set1 and set2


Your search resulted in set3     203 1*2

USER::
show set3 title docs1-3


1
LA-UR-75-1633   NTIS Prices: PC$3.50/MF$2.25
  Meson Radiobiology and Therapy
  Aug 75    8p

2
COO-2360-4   NTIS Prices: PC$3.50/MF$2.25
  Damage   and  Repair  in  Skin  Following  Exposure  to  Radioactive
Particles. Progress Report for the Support Period Ending 31 July 1975
  1975   8p
3
PB-246 283/6ST   NTIS Prices: PC$3.50/MF$2.25
  Methods  for  the  Production  of  Interferon  in  Cultures of Human
Diploid Cells
  See also PB-233 053.
  20 Feb 75   21p


USER::
nf skinrad
File has been opened.

USER::
view skinrad
skinrad contains        29 lines

USER::
lines 1-9


MEDLINE:
```

• • •

REOPEN FILE FROM PREVIOUS SESSION

REVIEW PART OF FILE (MEDLINE SDILINE SEARCH)

```
1
TI  -  ¢<Statistical evaluation of light protection factors¢>

2
TI  - Prognosis and post-therapeutic follow-up of breast cancers by

USER::
file
```

SAVE ONE TITLE FROM CURRENT SEARCH

```
USER::
show tit***set3 title docs2-2


2
COO-2366-4   NTIS Prices: PCS3.50/MFS2.25
  Damage   and  Repair  in  Skin  Following  Exposure  to  Radioactive
Particles, Progress Report for the Support Period Ending 31 July 1975
  1975   8p.

USER::
pick data 34


.FILE 34.
Event: Time,SearchTime,Date,User#,Descr,Docs,File
End:  8:04:38,015.23,01/19/76,0108,0002,0008,06
File reset: SCISEARCH 74-75 UK48

USER::
find radiation and skin


Your search resulted in set4      27 RADIATION(F)SKIN

USER::
show tit###set3 title docs1-3


1
  913965 (***not online***)

2
  913828  (***not online***)

3
  913168 (***not online***)
```

ERROR DUE TO ASKING
FOR SET NOT AVAILABLE
FOR CURRENT DATA BASE

```
USER::
pick data physics


.FILE12
Event: Time,SearchTime,Date,User#,Descr,Docs,File
End:  8:07:09,902.51,01/19/76,0108,0002,0000,34
File reset: INSPEC-PHYSICS 70-75 ISS 23

USER::
find radiation and skin


Your search resulted in set5      46 RADIATION(F)SKIN

USER::
```

show set5 title docs1-3

1
816773 A7574583
AN INTERCOMPARISON OF RADIOPHARMACEUTICAL KIDNEY KINETICS IN THE MOUSE

2
804602 A7567479
THE EFFECT OF IONIZING RADIATION ON PROTEIN METABOLISM IN STORED RAT SKIN

3
801027 A7563291
HEAT BALANCE AND THERMAL RESISTANCES OF SHEEP:S FLEECE

USER::
file

SAVE TWO TITLES FROM PHYSICS ABSTRACT SEARCH

USER::
show set5 title docs2-3

2
804602 A7567479
THE EFFECT OF IONIZING RADIATION ON PROTEIN METABOLISM IN STORED RAT SKIN

3
801027 A7563291
HEAT BALANCE AND THERMAL RESISTANCES OF SHEEP:S FLEECE

USER::
view skinrad
skinrad contains 47 lines

USER::
lines 30-47

REVIEW LAST 2 SEARCHES IN NTIS AND PHYSICS ABSTRACTS DATA BASES

2
COO-2366-4 NTIS Prices: PC$3.50/MF$2.25
Damage and Repair in Skin Following Exposure to Radioactive Particles. Progress Report for the Support Period Ending 31 July 1975
1975 8p
?

2
804602 A7567479
THE EFFECT OF IONIZING RADIATION ON PROTEIN METABOLISM IN STORED RAT SKIN

3
801027 A7563291
HEAT BALANCE AND THERMAL RESISTANCES OF SHEEP:S FLEECE

USER::
find skin

Your search resulted in 2875 SKIN

APPENDIX B

CONIT INSTRUCTIONAL MESSAGES

This appendix lists the various instructional messages a user can request currently in the experimental CONIT system. The initial "welcome" message a user sees when first entering CONIT is shown first to provide some context. Following that is listed the response to various instruction requesting commands. These commands include the 'help' command and the 'explain' command. As explained in Section 2.1, the 'explain' command name (abbreviation: 'e') is followed by the name of some concept the user wishes the CONIT system to explain.

The listings are copied directly from a computer terminal printout so as to give the format exactly as a user would see it, except that size has been reduced by 20 percent. Another exception is that the user commands have been underlined for clarity. The conclusion of a CONIT response is indicated by the user prompt (USER::).

It is to be pointed out that these messages represent our first attempt at an instructional dialog for CONIT. Experiment and analysis have suggested general principles and particular modifications we would propose, as indicated in the body of this report.

Welcome to CONIT. For help on how to use CONIT you may type 'help' followed by a carriage return; otherwise, you may now type any CONIT command.

USER::
help
In order to get information about how to use CONIT you type the command name 'explain' (or just e) followed by the name of the concept you want explained followed by a carriage return. For example if you type:
explain start
you will get an explanation of how to start to use CONIT. If you type:
e concepts
You will get a list of the concepts you can have explained.

USER::
e start
EXPLANATION OF GETTING STARTED
To get started you pick a retrieval system to search in by typing the command name 'pick' followed by the system you want. e.g. if you type:
pick medline
you will get connected to the MEDLINE system. Other systems you may pick are sdc,lms(Lockheed), and intrex.
After connecting to a retrieval system you may find out what data bases(files) are available by typing 'show data'.

USER::
e concepts
EXPLANATION OF CONCEPTS
To have a concept explained type 'explain' followed by a concept name
Some concepts you can have explained are:

CONCEPT NAME	CONCEPT
conit	Short introduction to CONIT system.
start	How to start running CONIT
commands	List of CONIT commands
converse	How user and CONIT converse: the mechanics of the interacting dialog.

Explanations are also available for the individual commands.

USER::
e conit
Explanation of CONIT
CONIT is a system that connects you to different information retrieval systems and allows you to select any database of any of these systems to search (find documents). You can use a common (CONIT) language for giving commands or use the language of the system you are searching in.

USER::
e converse
EXPLANATION OF HOW TO CONVERSE WITH CONIT
You talk with CONIT by giving it commands. Each command consists of a command name which may be followed by one or more additional words to make the meaning of the commands clear. (Type 'explain commands' for details.) To signal the computer that you have completed your command you MUST strike the carriage return key; the computer will just wait until you do.
CONIT will respond to your command with some message. To signal that its message is complete and that it is again waiting for your command CONIT will print the 'user cue': USER:: (just :: in terse mode). You cannot give a command until you get the USER:: cue but, if you are connected to CONIT directly thru MULTICS (not thru ARPANET) you can interrupt CONIT in its processing of its last command by typing the BREAK key after which CONIT will give you a USER:: cue.
For other details, like how to string commands together, type
explain converse more

USER::

e commands

EXPLANATION OF COMMANDS

The following is a list of CONIT commands.
Type 'explain X' , where X is command name, for further explanation.

NAME	ABBREV*	SHORT EXPLANATION
explain	e	Explain CONIT concepts.
pick	p	Pick a retrieval system and database to search.
find		Search database to find documents.
show		Show information on documents, data bases, etc.
combine		Combine sets of retrieved documents.
speak	sp	Change command language or language mode.

ABBREV* = abbreviated form of command.

USER::
e pick

EXPLANATION OF PICK COMMAND

The PICK command is used to pick systems and data bases to search in.
To pick a system type 'pick X', where X is the name of the system; e.g.,

pick medline

will get you connected to the MEDLINE system. Other systems you may pick are sdc, lms (lockheed) and intrex.

To pick a data base type 'pick data X', where X is data base name; e.g.,

pick data ntis

will set the NTIS data base up as the current one you can search (if it is available). Type 'show data' for list of data bases that should be currently available from the currently connected system.

USER::
e find

EXPLANATION OF FIND COMMAND

The FIND command is used to search for documents indexed under a particular term. Type 'find X' where X is the term you are searching for. For example,

find transporation
find radiation effects
find energy conservation

If you want to know what are alphabetically nearby terms under which documents are posted type 'show index X'.
For further information on how to make particular searches, type 'explain findmore'.

USER::
e show

EXPLANATION OF SHOW COMMAND

The show command gives information about documents that have been found in searching, about data bases, about index terms to search on, etc.
To have CONIT show standard citation information on some of the last set of documents you have found just type 'show '; you may also be more specific:

show set3 title docs1-4

will cause the titles of the first 4 documents of set3 to be shown to you.
For more details on how to get document information type 'explain show docs'
Examples of other information that can be obtained are given below:

COMMAND	BRIEF EXPLANATION
show data	Lists data bases currently available
show systems	Lists systems currently available
show index X	Lists index terms alphabetically near X
show news	Gives news from connected system

For more details type 'explain show data', etc.

USER::

'e combine
EXPLANATION OF COMBINE COMMAND
The COMBINE command allows you to make Boolean combinations of the sets of documents you have prieviously found from searching your currently connected data base; for example,
combine set2 and set5
will make a new set which contains only documents which are in both set2 and set5. Similarly,
combine set2 or set5
makes a new set with all documents from either set2 or set5. Also,
combine set2 and not set5
will make a new set which contains documents in set2 but not in set5.

USER::
e speak
EXPLANATION OF SPEAK COMMAND
The speak command allows you to change the command language you are using to speak to the currently connected system. Initially, the CONIT language is setup. When you are connected to some host system you may speak to it in the common CONIT language or in the host language. To speak in the host language type:
speak host
When you are speaking in host language no regular CONIT commands will be recognized except one:
speak conit
which resets the language to CONIT.
When speaking in the CONIT language you will get explanatory messages from CONIT. After you become familiar with CONIT you may want to have these instruction messages shortened. You request this by typing
speak terse
To resume more lengthy explanations type
speak verbose

USER::
expalin sunynews
expalin is not a legal CONIT command.
Type 'explain commands' for a list of commands.

USER::
explain sunynews
News is not available from the SUNY/MEDLINE system; to get news about all the MEDLINE systems, including SUNY, 'pick medline' and 'show news'.

USER::
explain news
CONIT cannot yet explain news
Type 'explain concepts' for a list of concepts Conit can now explain.

USER::

APPENDIX C

CONIT TRANSLATION TABLES

This appendix lists the five pairs of translation tables regularly used in the CONIT system. Tables are listed for systems in this order:

1. NLM MEDLINE (tag: med)
2. Lockheed DIALOG (tag:lms)
3. Systems Development Corporation ORBIT (tag:sdc)
4. M.I.T. Intrex (tag:intrex)
5. SUNY MEDLINE (tag:suny)

For each system first the command translation table (see Section 2.4) and then the response translation table (see Section 2.3) is given. The tables are reproduced from computer listings. The commands set_table (abbreviated:st) and list_table (abbreviated:lt) are used to make the table operative and then to list it, respectively (see Section 2.5.1). The tables themselves have been boxed in and labeled for ease of viewing in this report. Note that each entry starts at the left hand margin and ends with the asterisk (*) -- spaces are important. The input (left-hand or argument) side of each entry is separated from the output (right-hand or translation or function) side by the equals (=) sign.

NLM MEDLINE COMMAND TRANSLATION TABLE

```
yes=YES*
tohost=*
title=TI,*
show systems=e show.systems*
show news="NEWS*
show more=down 5*
show index="IBR*
show data all=""FILES*
show data="FILES?*
show="PRINT*
pick data="USERS""FILE*
offline=OFF-LINE*
ls all=ls all*
logout="STOP"*
find author="FIND*
find="FIND ALL*
docs1-=*
combine set=*
abstract=AB,*
+=:*
show systems= show.systems*
 show news= show.news*
 show index= show.index*
 show docs= show.docs*
 show data= show.data*
 show= show*
 set= SS,*
 or set= OR *
 find more=  find.more*
 find= find*
 converse more= converse.more*
 and set= AND *
 and not set= AND NOT *
 all= DETAILED*
```

USER::
set_table out med

USER::
list_table out

NLM MEDLINE RESPONSE TRANSLATION TABLE

```
UP N OR DOWN N?=To see more type 'show more'.*
[redacted]=COMIT*   [ARGUMENT HERE IS USER ID; BLOCKED AND
SS (=Your search resulted in set*  TRANSLATED FOR SECURITY]
PROG:=MEDLINE:*
MISSING DOUBLE-QUOTE MARK.=*
) PSTG (=  which contains this many documents: (*
 /C?= is the number for your next MEDLINE search set.*
```

USER::

```
set_table lms

USER::
list_table
```

DIALOG COMMAND TRANSLATION TABLE

```
tohost=*
show set=t*
show offline set=print*
show offline=print1*
show news=?news*
show more=0*
show index author=eau=*
show index=expand*
show data=?files*
show=T1*
pick data=.file*
ls all=ls all*
find author=sau=*
find=s*
combine set=c*
+=?*
title=/6*
psychab=11*
physics=12*
er set=+*
or =e lmsor*
ntis =6*
eric=2*
eleccomp=13*
docs=/*
compendex=8*
citation =/2*
chemab=3*
cain=10*
and set=**
and not set=- *
and =(F)*
all=/5*
abstract=/4*
```

USER::

```
st out lms

USER::
list_table out
```

DIALOG RESPONSE TRANSLATION TABLE

```
type in LOGOFF as your last command,=You are spealing in COMIT.*
sure proper accounting of your Runtime,=*
a  ?  from the computer=the USER:: cue from COMIT.*
Type in LOGOFF as your last command,=*
Tel: (415)493-4275=Lockheed DIALOG*
Please call  (415) 4=*
LOGOFF at=DIALOG session terminated at*
DIALOG command=COMIT command*
93-4275 for questions or problems=*,
*** IMPORTANT... To in=*
 just prior to hanging-up the phone=*
  IT= documents in set for term *
  ?  f= USER:: prompt f*
     0  =:no documents found: try 'show index yourterm'*
           T=           T*.
                                  -more-=For more type 'show more'*
           =Your search resulted in set*
```

USER::

```
st sdc

USER::

lt
```

ORBIT COMMAND TRANSLATION TABLE

```
tohost=*
title=TI,*
show news="NEWS*
show index="INDR*
show data all="EXPLAIN SCHED*
show data="FILES?*
show="PRINT*
pick data="TIME""FILE*
offline=OFF-LINE*
ls all=ls all*
find author="FIND*
find="FIND ALL*
docs1-=*
combine set=*
abstract=AB,*
+=:*
 set= SS *
 or set= OR *
 and set= AND *
 and not set= AND NOT *
 all= FULL*
```

```
USER::

st out sdc

USER::
lt out
```

ORBIT RESPONSE TRANSLATION TABLE

```
PROG:=SDC/ORBIT:*
NP (OPEN)=Connection completed.*
MISSING DOUBLE-QUOTE MARK.=-.*
/C?=is the number of your next SDC/ORBIT search set.*
```

```
USER::
```

```
set_table intrex

USER::

list_table
```

INTREX COMMAND TRANSLATION TABLE

```
show=output*
or set=or s*
find title=title*
find author=author*
find=subject*
combine set=s*
and set=and s*
and not set=and not s*
```

```
USER::
```

```
set_table out intrex
USER::

list_table out
```

INTREX RESPONSE TRANSLATION TABLE

```
your request TITLE=your request FIND TITLE*
your request SUBJECT=your request FIND*
your request AUTHOR=your request FIND AUTHOR*
named s=named set*
current list=current set*
```

```
USER::
```

```
st suny

USER::
lt
```

SUNY MEDLINE COMMAND TRANSLATION TABLE

```
tohost=*
title=TI,*
show news=explain sunynews*
show index="NBR*
show data all="FILES*
show data="FILES?*
show="PRINT*
pick data="USERS""FILE*
offline=OFF-LINE*
ls all=ls all*
find author="FIND*
find="FIND ALL*
docs1-=*
combine set=*
abstract=AB,*
+=:*
 set= SS *
 or set= OR *
 and set= AND *
 and not set= AND NOT *
 all= DETAILED*
```

```
USER::


st out suny

USER::
list_table out
```

SUNY MEDLINE RESPONSE TRANSLATION TABLE

```
SS (1) PSTG (168)=Connection completed.*
SS (=Your search resulted in set*
PROG:=SUNY/MEDLINE:*
MISSING DOUBLE-QUOTE MARK=*
EDTST05=COMIT*
HOST SYSTEM=SUNY/MEDLINE*
) PSTG (= which contains this many documents: (*
 /C?= is number for your next search set.*
```

```
USER::
```

APPENDIX D

SUGGESTED USER PROTOCOLS FOR ACCESS TO A COMPUTER SYSTEM VIA A NETWORK

It is noted that a bibliographic retrieval system is frequently accessed through networks that also provide access to other retrieval systems and systems providing service in other application areas. In this appendix we suggest some procedures by which networked access to retrieval systems and other systems may be standardized for users.

We start from the assumption that access to several current bibliographic systems should require only two pieces of information from the user: name of service desired and user's password, which implies his identification. An attempt has been made in the suggested protocols to be be compatible with, or adaptable to, different terminal types, more general functional requirements (other than access, per se) in the retrieval application, more general application areas, and developing common or virtual system approaches.

The "standards" we propose need not imply a system must have all functions to be standard. Rather, they say, for example: an EXPLAIN function may be a good facility to have and, if you have it, here is the standard way it should appear to the user. Similarly, on request for service: if the service is implied by the physical connection, there is no need to insist on the request; but if there is a request, here is the standard protocol.

PROTOCOLS

1. System (Network) Acknowledgement and Service Request (After establishment of telephone connections and terminal speed and type identification)

   HELPFUL NETWORK (1) 13:45 EST (2) 76-1-31 (3) (617) 964-2007 (4)
   REMEMBER NEW PHONE NUMBERS NOW AVAILABLE (5)
   TYPE (6) NAME OF SERVICE YOU WANT (FOLLOWED BY CARRIAGE RETURN) (7) :: (8)

NOTES

(1) Name of acknowledging system given here. Encourage client systems of networks to allow this (or at least some identifying phrase) to make it easier for user (and system analysts) to know what's going on.

(2) Time

(3) Date "Standard" -- year-month-day -- order used; however, it should not be necessary to force non-suppressed zeros on user if hyphens are given as separators).

(4) This is telephone number of port connection and can serve as check for caller and as useful debugging device to identify bad lines, modems, etc. Alternate form: BOSTON PORT 7.

(5) Optional system message of day.

(6) "TYPE" is more explicit than, e.g., "ENTER".

(7) This phrase may be needed for inexperienced user if good timeout and recovery is not available (see below). If NEW LINE becomes established in place of carriage return, some change will be needed.

(8) System signal is two colons followed by carriage return. (See Section 4.3)

2. Service Request Response (by User)

   orlog (1)

NOTES

(1) Name of service given here.

(2) User responses should be allowed in either upper or lower or mixed alphabetic cases.

3. Service System Acknowledgment and Password Request

THIS IS ORLOG (1)

TYPE YOUR PASSWORD:: (2)

NOTES

(1) We assume entry of a correct service name by user causes control to be passed to service system.

(2) We assume here a non-print mode is now entered for password security. If non-print mode is not available, the format would be:

TYPE YOUR PASSWORD

:: YOUR PASSWORD

The two colons would be the last part of the message and the next typing position to the immediate right of the colons which is the first character in a string of underprinting characters which mask the password. The underprinting string would start with the string "YOUR PASSWORD" and be overprinted by two or more additional lines to assure masking (this device works well on MULTICS).

4. Service System Password Acknowledgment

Welcome to ORLOG...

NOTE

This message implies acceptance of password.

5. Alternate Multistatement Request Response (User/System)

login (1) orlog (2) :: (3) pass (4) xxxxx (5) select eric find ... (6)

NOTES

(1) "login" (synonyms: log or l) is command name which tells system that user wants to get out of response-limited mode and string together several components of access procedure in one statement.

(2) First argument is service name (but see (4))

(3) As an exception to general rule for user/system signals, two colons (no carriage return) are sent by system to indicate that service name is recognized and control passed on. Possibly, this signal could be eliminated in a variant command, say "logon."

(4) Second argument (synonyms "password" or "p") indicates next word is password. This argument could be assumed by content but an explicit indication might be easier to implement in the more general system-to-system interconnections. For the same reason it might be better to insist on an identifying argument (say "service") preceding service name.

(5) Password with appropriate non-print or non-print or other security measures. Security measures should probably be responsibility of network.

(6) Optional additional commands sent to service system.
NOTE: All user input after service name would be passed along to service system and allow for indefinitely long "batch" operations.

6. Error Message: Invalid Service Name

ORLOX (1) is not a valid service name. If you want to see list of available services, type LIST: (2) Otherwise, type name of service ::

NOTES:

(1) Incorrect service name feedback to user.

(2) This CAI option should be allowed where knowledge of services is not restricted.

7. ORLOX (1) is the third (1) successive invalid service name we have received. Call HELPFUL NETWORK representative for help at xxx-xxxx(2). Your terminal connection is now being dropped (3).

NOTES

(1) Three strikes and you're out!

(2) Telephone number to call for help.

(3) Tell user he's being disconnected.

8. Error Message: Invalid Password

Incorrect password received. To see what was received, type ECHO(1) otherwise, re-type password::

NOTE

(1) This option should be available when user can accept lack of security in printing of near-password.

9. Error Message: Repeated Invalid Password

Incorrect password received three successive times. Call your ORLOG representative for help at xxx-xxxx. You are being returned to HELPFUL NETWORK. [Followed by Message 1.]

10. Timeout Message

No response received in 2 minutes. Call HELPFUL NETWORK representative if you need help: xxx-xxxx. Your terminal connection is now being dropped.

11. Exit Commands

11.a login helpful.

NOTE

This means that the user wants to leave service system (after appropriate exit and accounting messages) and return (or login) to system indicated. This may require passing appropriate information to other systems. Default condition (no argument: synonym EXIT) would mean drop back to calling system.

11.b Logout.

NOTE

This means user wants to stop altogether and have his terminal disconnected. Logout is the natural antonym for login.

12. Edit Commands

12.a Cancel m preceding characters: m "left arrows".

NOTES

(1) Buffered terminals can replace deleted characters

(2) Is it important to use an ASCII character?

12.b Cancel line: @ (m at-signs would mean cancel last m lines)

13. Interrupt

Requested through special button (normally activating a line-condition change rather than a character -- called INTERRUPT, ATTENTION, QUIT, etc.), a very important function, especially on system output where it means "stop output, give system signal, and allow user input." On user input it can be used instead of cancel line (system signal invoked).

14. CAI Commands

14.a Explain x (synonym: exp).

Argument may be message name or word in message, for example. May be useful even in access procedures, as in "explain service" which might, along with some other explanation, do same as "list" (see (6)). Default condition: explain last system message further.

14.b Help (synonym: ?).

Generalized CAI for current context; i.e., what current user options are and how to get further information on those options.